12

QUARTERLY TECHNICAL REPORT ON THE EARTH STATION INTERFACE FOR DEMAND ASSIGNMENT APPLICATIONS

FOR THE PERIOD 1 JANUARY 1982 THROUGH 31 MARCH 1982

Sponsored by:

Defense Advanced Research Projects Agency

ARPA Order Number 3674/5

and

Defense Communications Agency

Contracting Agency: DSSW

Contract Number MDA903-81-0638

Effective Date of Contract: 81 SEP 08

Expiration Date of Contract: 82 DEC 31

JOHN E. McCORMICK
Program Manager
(714) 453-7007 X577

"The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U. S. Government."

M/A-COM LINKABIT, Inc.
10453 Roselle Street
San Diego, CA 92121

During the first quarter of calendar year 1982, the principal ESI design work was directed toward circuit card assembly (CCA) design. Two CCA's representative of the work performed are the Control and Interface Processor (CIP) for the Interface Control and Codec Unit (ICCU) and the Preamble Detection board located in the Modem.

The CIP design is actually common to two assemblies which differ principally in software. One is used in the uplink, the other in the downlink. These two processors comprise the heart of the ICCU traffic control system.

The Preamble Detection board is a brand new design which will make substantial performance improvements in the old design. The old design was implemented with discrete I.C.'s utilizing a recursive algorithm. The new design uses LSI digital correlators in a non-recursive operation which operates at higher speeds with less probability of errors.

The two discussions are presented as individual high level reviews.

M/A-COM LINKABIT, Inc.
3033 Science Park Road
San Diego, CA 92121

CONTROL AND INTERFACE PROCESSOR

28 April 1982

By: Tom Fletcher

List of Figures

CONTROL AND INTERFACE PROCESSOR

INTRODUCTION

1. The LCC36A (Interface Control and Codec Unit) has two Control and Interface Processor (CIP) cards. These cards have been redesigned to be identical with respect to the hardware. Only the software will distinguish between the uplink CIP and the downlink CIP.

2. FUNCTIONAL DESCRIPTION

2.1 THE UPLINK CONTROL AND INTERFACE PROCESSOR (CIP)

The uplink CIP is responsible for coordinating the flow of traffic generated by the PSAT. This includes both information sent over the satellite network and local control information.

The traffic that is intended for the network is passed to the encoder which is located in the LCC36A. That information is then sent to the modem (LBM36A) which is initiated and controlled by the uplink CIP.

When local control information is received, whatever action to be taken is initiated by the uplink.

2.2 THE DOWNLINK CIP

The downlink CIP is responsible for coordinating the flow of traffic received from the Decoder to be transmitted to the PSAT. This includes both incoming information from the satellite network and local control information.

The incoming network traffic is demodulated by the LBM36A modem and then passed to the decoder in the LCC36A. The decoded data is then passed to the downlink CIP for eventual transmission to the PSAT.

When local control information is received, whatever action to be taken is coordinated with the uplink processor and information is sent to the PSAT as necessary by the downlink processor.

3. DESIGN APPROACH

The general design philosophy is to upgrade the functionality of the processor boards with as little impact to the current hardware and software as possible.

3.1 OBJECTIVES AND REQUIREMENTS

The following is a list of upgrade objective and requirements.

- Support downloading from tape cassette.
- Support an RS-232 interface.
- Control BIT on peripheral boards.
- Make the uplink and downlink processor boards compatible with the LCC36 units.
- Replace the currently used PROM and RAM with 64K words of dynamic RAM for system memory.
- Replace the existing DMA Controller circuitry with LSI-DMA Controller chips.
- Implement a Modem message interface using an Intelligent Peripheral Controller.

4. MODULE DESCRIPTIONS

4.1 DMA

The DMA Controller shares the system bus with the MC68000. All DMA transfers (except interprocessor DMA) are performed via this bus. There will be 3 DMA channels per board, i.e.:

<u>UPLINK PROCESSOR</u>

1. Memory ---> Encoder

2. PSAT ---> Memory

3. Memory ---> Downlink Processor

DOWNLINK PROCESSOR

1. Decoder ---> Memory

2. Memory ---> PSAT

3. Memory ---> Uplink Processor

The PSAT and CODEC DMA channels will operate in basically the same manner as they have in the past. The only difference will be the need for software initialization of the controller, and a different chaining scheme. Obviously, the basic requirement for an address and a word count remain. Specific implementation details will depend upon the controller chosen.

4.1.1 HARDWARE IMPACT

The LSI-DMA controllers replace approximately 40 IC packages. These ICs make up the discrete components of the current controller hardware.

Although great care was taken to select and design hardware which would not affect the system software, some changes will be necessary. Obviously, the software will have to perform the controller initialization and programming. In addition, the chaining scheme will be slightly different. Specific changes will depend on the controller used, but the general operation of the software with respect to DMA will remain unchanged.

Consideration was given to 2 products. The Motorola MC68450 DMA Controller and the AND AMZ8016. Although the Motorola controller seems well suited to our needs, it will not be

- TRANSFERS CAN BE M——M, P——P, M——P
- UP TO 1.3M BYTES/SEC
- TWO-STEP OPERATION
  1 SRC——TEMP
  2 TEMP——DEST

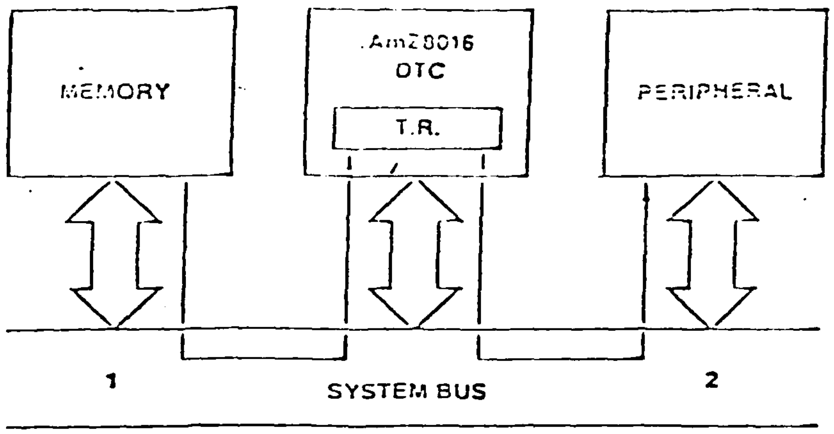
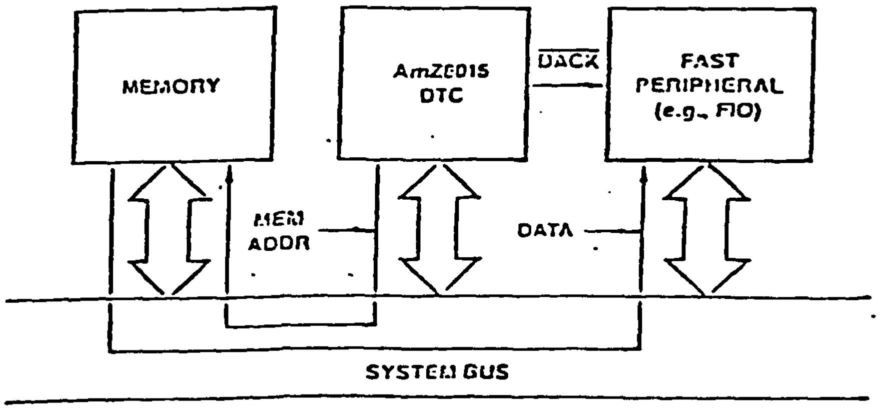


Figure 1 : Configuration of Flowthru Transaction

Figure 2 : Configuration of Flyby Transaction

- ONE STEP TRANSFER
- DTC ADDRESSES MEMORY. $\overline{\text{DACK}}$ SELECTS PERIPHERAL. DATA TRANSFERS M——P
- M——P ONLY. UP TO 2.0M BYTES/SEC
- $\overline{\text{DACK}}$ IS NOT A BUS SIGNAL



available until te 3rd or 4th quarter of 1982. Therefore, we have selected the AMZ8016. Current and proposed applicaticns will require four DMA channels per board. This means that two Z8016 packages will be used. Some useful Z8016 features are:

1. Two Channels (independent).
2. Transfer Modes: Single, demand dedicated with bus hold, demand dedicated with bus release, demand interleaved.
3. Optional automatic chaining of operations.
4. Byte or word operands.
5. Memory/peripheral transfer up to 2.66 Megabyte/sec.
6. Byte packing/unpacking.

This device operates on a 4 MHz clock, and can transfer a word in 3 clock cycles in FLYBY mode (Figures 1 & 2). This implies that the Z8016 can transfer 1.33 Mwords/sec (2.66 Mbytes/sec). Current DMA transfer requirements are 1 M byte/sec, so the Z8016 is more than adequate.

4.2 DYNAMIC RAM

Currently, system memory consists of 10K words of PROM and 5K words of RAM. One of our redesign objectives is to replace this memory with 64K words of Dynamic RAM. This will give us a greatly expanded memory with fast access. With the exception of a bootloading PROM, all other PROMs will be replaced by Dynamic RAM. This will eliminate the PROM reprogramming every time a system software update is made. Whenever a software update is released, or a system failure has caused the program code to be lost, the new code will be loaded into the memory via a simple Downline Loading program residing in the Bootload PROM.

The nature of Dynamic RAM is such that its contents must be periodically "refreshed" or else the stored information will be lost. This requirement creates an additional processing task. This task is performed by a Dynamic Memory Controller (DMC) in conjunction with MC68000 and some Timing/Control circuitry.

The dynamic RAM configuration diagram (Figure 3) shows the significant features of the proposed Dynamic memory system. This functional module consists of four major components: the timing and control unit, memory controller (DMC), 64K word memory bank and parity generation/check circuitry.

The timing and control unit (TCU) generates the appropriate input signals to the DMC as a function of the MPUCLK and the bus control signals to ensure a timely refresh sequence.

The DMC generates the proper row and column address and control signals as a function of the address bus and the TCU

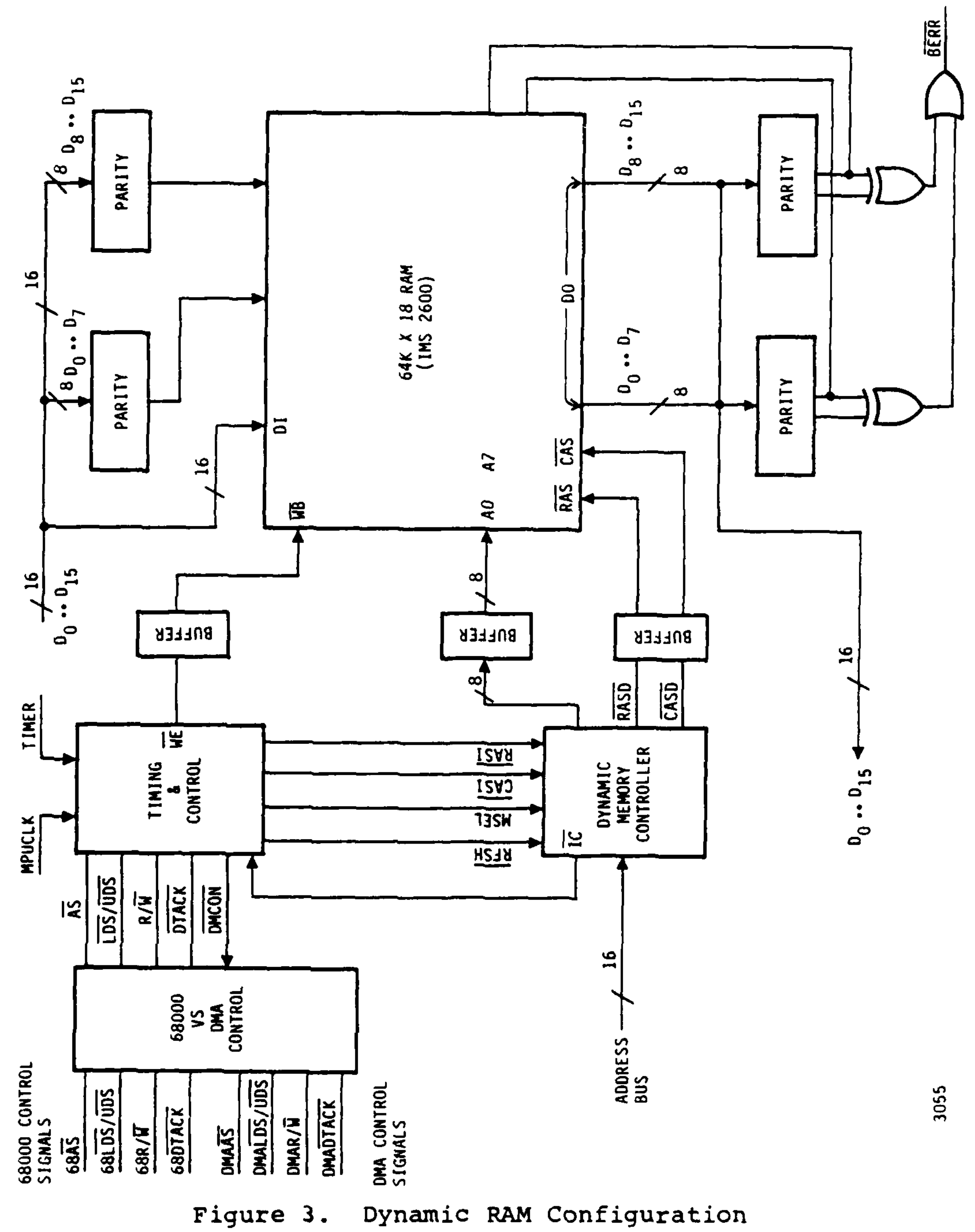


Figure 3. Dynamic RAM Configuration

outputs. The DMC has an internal row address counter/generator that may be used during refresh cycles.

The error detect circuitry computes a parity bit for each byte during each write to RAM. It also checks parity whenever the RAM is read. Parity errors will generate a bus error. This will invoke the appropriate exception code.

The 64K word dynamic memory bank is organized as 256 rows by 256 columns. The RAMs (IMS2600) have a 190ns access time. Each row in the memory bank must be refreshed (accessed) at least once in every 4ms intervals.

The basic refresh philosophy is one of performing the refresh operation between bus cycles, and if necessary, at the beginning of the bus cycle. These states are detected from the $\overline{\text{AS}}$ signal generated by the 68000. When $\overline{\text{AS}}$ goes inactive we are either between bus cycles, or right at the beginning of a bus cycle. $\overline{\text{AS}}$ remains inactive from the beginning of a bus cycle until the time that the address bus goes active. We can take advantage of this time lag to finish the refresh operation; after which, the bus cycle should proceed in a normal manner. Preliminary examination suggests that we may have enough time to perform a "hidden refresh" with an8 MHz clock. If we don't have enough time at the beginning of a bus cycle to perform a "hidden refresh", we could do something like stretching the bus cycle by delaying the $\overline{\text{DTACK}}$ (Data Transfer Acknowledge) signal. In order to ensure that the hardware will operate at higher speeds (10-12 MHZ), this $\overline{\text{DTACK}}$ control will be implemented even though it may never be active at lower speeds. A "normal" refresh is defined as a refresh which occurs between bus cycles. (i.e., $\overline{\text{AS}}$ is inactive, and it is not the beginning of a new bus cycle). If $\overline{AS}$ is still inactive after a normal refresh cycle (i.e., we are not in a bus cycle) another refresh is executed. This continues until we run into a bus cycle or have performed 256 refreshes. If we run into a bus cycle, the current refresh cycle is completed (Hidden Refresh), and refresh operations are temporarily suspended until $\overline{AS}$ again goes inactive (End of bus cycle). The DMC will tell us when we have performed 256 refreshes via a $\overline{TC}$ (terminal count) output. Thus, refreshes will be stopped after $\overline{TC}$ goes active until a 2ms timer starts the entire memory refresh cycle over again. A write protection will be implemented. This protection is enabled immediately after downlink loading to insure the system code cannot be written over. It can only be disabled during a software reload.

4.2.1 HARDWARE/SOFTWARE IMPACT

The use of Dynamic RAM has little or no effect on existing software. However, there will be a need for some additional software tools (i.e.,Bootloading and Down-line loading routines). Unlike the software, hardware is going to be significantly affected. PROMs will be replaced by DRAMs, a DMC will be added, a timing and control unit will be added, parity generation/ checking hardware will be added and a tape drive (for software backup and reloading) will be added.

4.2.2 ALTERNATIVES

4.2.2.1 DYNAMIC MEMORY CONTROLLERS

The Dynamic Memory Controller selected is the AM2964B. An alternative controller considered was the National Semiconductor DP8409. The selection of the AMD device was based upon simplicity and functionality factors.

4.3 MODEM - ICCU COMMUNICATION

The uplink processor presently uses a port to transfer messages to the modem. The messages are variable in length and are transferred a byte at a time. The 8 bit output port is supported by a PIA type device (Figure 4). When the processor has something to send, it determines if the PIA can accept a byte, and if so, strobes it in. On the other end, the modem tells the PIA when it can accept a byte and the PIA generates a strobe to the modem if it has a byte to send. At this point, the

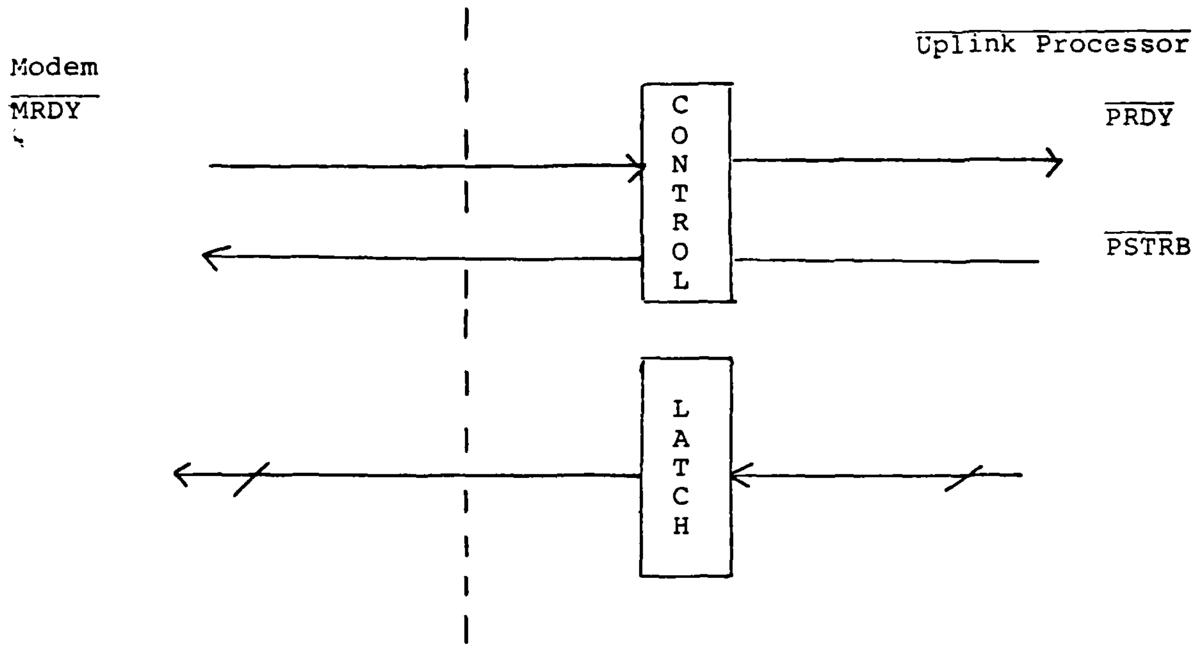


Figure 4. Current Modem - ICCU Interface

PIA is ready for a new byte from the processor.

The processor uses a circular software buffer to queue messages to the modem. This buffer is added to as messages are queued and deleted from as bytes are transferred to the PIA. The processor is responsible for all buffer management.

The downlink processor presently uses a port to accept messages from the modem. This port is also 8 bits wide and supports a PIA-type protocol in a manner analogous to the uplink. The downlink processor queues messages received from the modem in a circular buffer and is responsible for all the buffer management.

The downlink processor receives 2 distinct types of messages from the modem. One type contains modem control information and the second type contains T & M data. These messages require separate processing. Presently, the downlink processor searches for T & M messages when it takes data from its circular buffer and when found they are put in a separate T & M buffer and processed later on.

Both the uplink and downlink processors use circular buffers for transferring messages to and from the modem and are responsible for both placing data in these buffers and taking data from these buffers. One can envision a special port to circular buffer controller that might share these responsibilities. (Call it a PCB controller). Both the PCB controller and the processor would share the circular buffer control parameters. These would include a pointer to the beginning of the buffer, a pointer to the end of the buffer, and the buffer length. The PCB would have access to the buffer over the system bus.

Consider how this might work on the uplink. As the uplink processor had messages to send it would put them in the circular buffer and alter the buffer pointer and byte count accordingly. The PCB would monitor the modem's $\overline{\text{RDY}}$ line and transfer data from the circular buffer to the modem when the modem could accept data. As the PCB took data, it would modify the buffer pointer and byte count accordingly.

The downlink would operate in an analogous fashion. However, in order to support a separate T & M queue, 2 sets of buffer pointers and lengths, as well as 2 circular buffers, would have to be shared (i.e., our PCB controller would be used in conjunction with our T & M prefilter. See Figure 5).

The programmable I/O port would permit the same hardware to be used for both the uplink and downlink boards. This port, as well as the PROM code required by the PCB, would be on a local bus separated from the system bus. The PCB and the 68000 would share buffers and buffer pointers in common RAM.

4.3.1 HARDWARE/SOFTWARE IMPACT

Implementation of a message processor on the Modem - ICCU communication interface will take some of the message processing responsibilities away from the main processor. This will eliminate the need for certain sections of code in the 68000's software, and allow the 68000 to spend more time on its primary function.

The hardware impact is significant. The PIA would be eliminated, but in its place will go an intelligent peripheral processor, Local EPROM, and hardware to support a shared bus/memory scheme with the main processor. The critical requirement here is to keep time on the system bus to a minimum.

The Motorola 6812 peripheral controller has been selected for implementation in this design.

4.4 DOWN-LINE LOADING

Another redesign objective is the implementation of down-line loading capabilities. Two separate techniques have been implemented. One would support down-line loading via the RS-232 interface, and the other via a minicartridge (tape) drive. Either technique can be used for making software updates or restoring software after a system failure.

A Burroughs TM110 minicartridge drive has been selected for

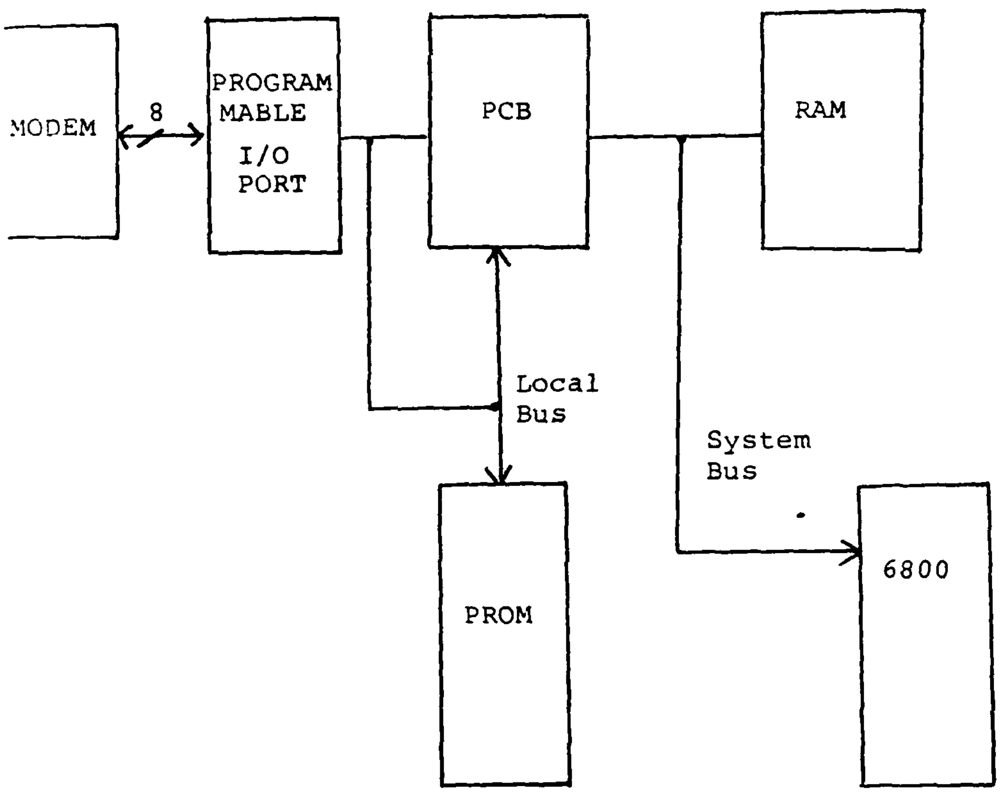


Figure 5. Peripheral Controller Configuration

use as the system tape drive. It was primarily chosen because of its size and simplicity. Data will be stored on this tape in one of two ways. A tape can be created in-house or it can be written to by the processor from information received over the satellite link. Data is read off of the tape whenever a software reload is initiated. Inputs to and outputs from the tape drive go to both processor boards. Drive control will be arbitrated by select circuitry in conjunction with interprocessor communication. This means that a processor wishing to use the drive must request it from the other processor if it doesn't "own" it. If the processor in Control is not using the drive, it will honor the request and disable its drive interface and controller. Otherwise the request will be queued until the "owner" gives up the drive.

The tape drive controller will be implemented using the Motorola 68121 with software performing drive monitoring, control, and data sampling. Data on the tape is recorded in phase-encoded form.

An RS-232 Interface for communication with either of the processors is to be implemented. An arbitration circuit similar to the one used for the tape drive will be used. A certain processor will "own" the interface. To change the operational context to the other processor, a CHANGE CONTEXT command must be issued to the "owner" processor. The "owner" will then signal the other processor (via the interprocessor communication Link) to assume "ownership" of the interface (as it relinquishes ownership). This technique makes context switching as simple as a keystroke.

Downline loading will be a simple operation through this interface. The only thing that needs to be implemented is a software command to perform the actual data load. Other useful commands will be, Memory-to-Tape Load, Tape-to-Memory Load and Interface-to-Tape Load and Context Identify.

4.4.1 HARDWARE/SOFTWARE IMPACT

A minimal amount of additional control hardware will be needed. A software tape drive controller will be required along with down-line loading code.

4.5 TIMING FACILITIES

An LSI Counter/Timer chip will be used to generate the timing control signals used by the various modules of the processor board. Timing signals are required by the tapedrive controller, the BIT unit, the RAM controller and the USART.

The timer selected for use on the processor board is the AMD AM9513 System Timing Controller. It was selected for two basic reasons. First, it has more timers (5) than other packages. Secondly, this particular timer is also used in the LBM36A.

4.5.1 HARDWARE/SOFTWARE IMPACT

The current timer will be replaced by the AM9513. No additional circuitry will be needed. The software which initializes and controls the timers will be changed to operate with this new device.

4.6 BUILT IN TEST

All peripheral boards in the ESI will incorporate a processor driven signature analysis type of Built In Test (BIT). This entails artificially stimulating the peripheral board with known inputs stored in board-resident PROM, and then evaluating selected board outputs by calculating their signatures and comparing them to known signatures stored in processor memory as shown in Figure 6.

It is assumed the ESI will not be connected to the channel whenever BIT is initiated. The processor will be running in a test mode which will drive the BIT and there will be no PSAT or Channel message traffic. The processor-to-BIT interface will consist of a Control port, a status port and a signature port. The control port will enable the BIT unit, perform BIT clocking, strobe the BIT output latches and reset the BIT unit. The status port will indicate when a signature has been computed and the signature port will contain the signature. Note: All BIT clocking signals will be inactive during normal operation.

THE BIT-TO-PERIPHERAL BOARD INTERFACE

The BIT-to-Peripheral board interface will be implemented as shown in Figure 6.

BITCLK is the processor provided built in test clock. It is used to synchronize the board under test to the processor signature analyzer.

BITSYNC is a synch pulse generated by the processor. It should be used to initialize both built in test circuitry on the board as well as the boards functional circuitry.

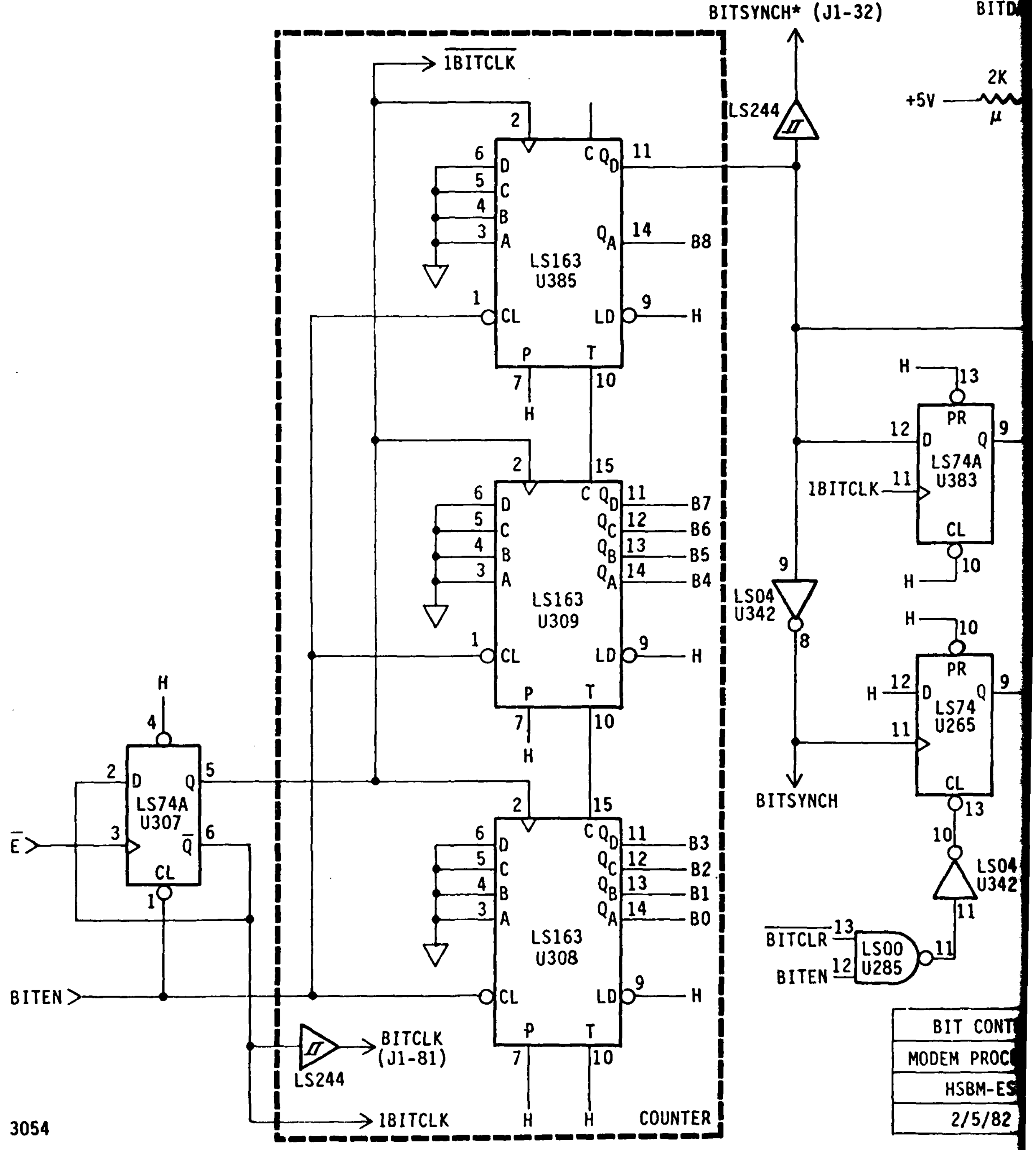
BITDATA is the output data on which the signature is being calculated.

BITDIS is a built in test disable signal.

BITSTRB strobes the built in test information sent via the processor data bus into the board under test.

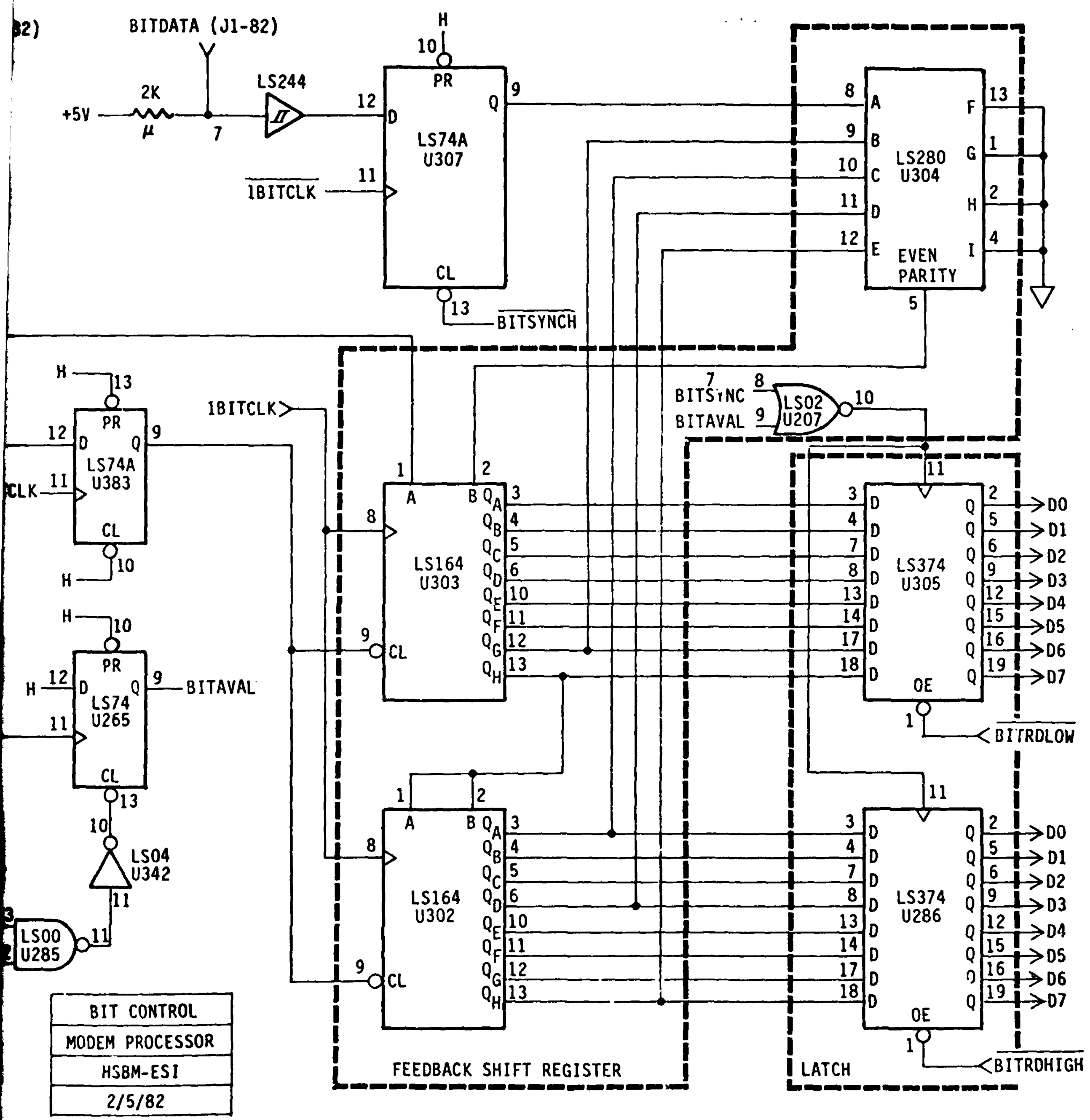
4.6.1 HARDWARE/SOFTWARE IMPACT

None of the built in test hardware exists on the current boards. BIT units will be added to the processor boards and the BIT support hardware will be added to the peripheral boards (Figure 7). A BIT support routine will be added to the system software.



BIT CONT
MODEM PROC
HSBM-ES
2/5/82

3054



BIT CONTROL
MODEM PROCESSOR
HSBM-ESI
2/5/82

Figure 6. BIT Control

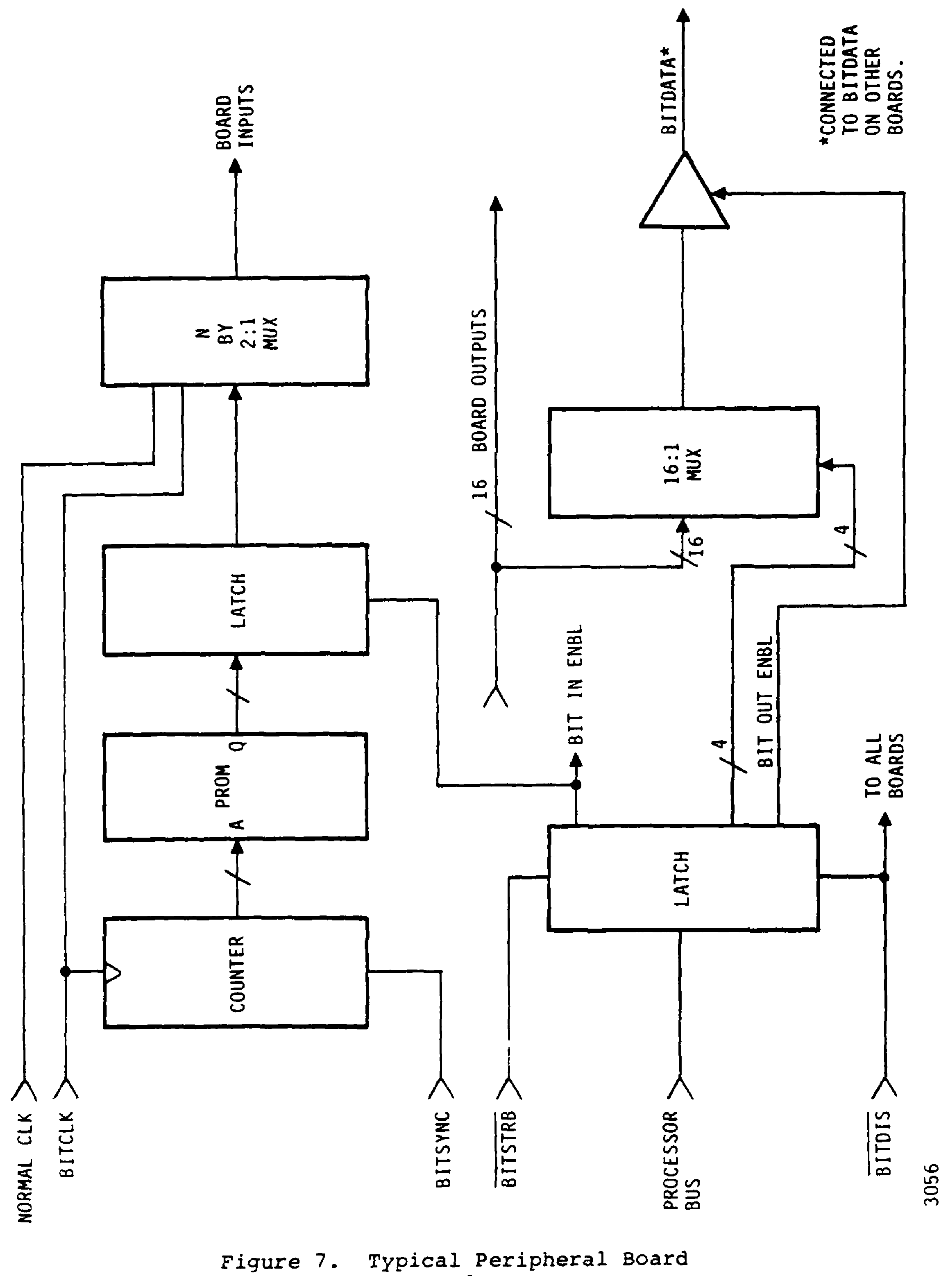


Figure 7. Typical Peripheral Board BIT Hardware

M/A-COM LINKABIT, Inc.
3033 Science Park Road
San Diego, CA 92121

PREAMBLE DETECT REDESIGN HIGH LEVEL DESIGN

5 May 1982

BY: Jim Petranovich

INTRODUCTION

The preamble detect board provides detection and initial estimation functions for the High Speed Burst Modem (HSBM) of the Earth Station Interface (ESI). It also provides prefilter functions for data processing.

A working preamble detect design is now in existence, but a redesign is to be implemented to simplify debugging and testing and to increase functional capability.

FUNCTIONAL DESCRIPTION

The preamble detect board accepts partially demodulated data from the receive filters and it performs a prefilter operation to generate data at the appropriate receive symbol rate. The filtered data is available both to the correlation circuit and externally to the DTxRx (Digital Transmit/Receive/Control) board. Its system configuration is shown in Figure 1.

The preamble detection process is performed on the filtered data. Coherent correlation with a known 32-bit sequence is performed independently on the inphase and quadrature channels, at twice the baud rate. A 32-bit energy calculation is also performed on the same data, at twice the baud rate.

The correlation can be compared with the energy to determine detection of a burst, if no burst is present and detection is enabled. The correlation can instead be compared with a fixed threshold for detection. Carrier detection can be achieved by comparing the energy with a fixed threshold. Burst and carrier detection are two basic outputs of the system. See Figure 2.

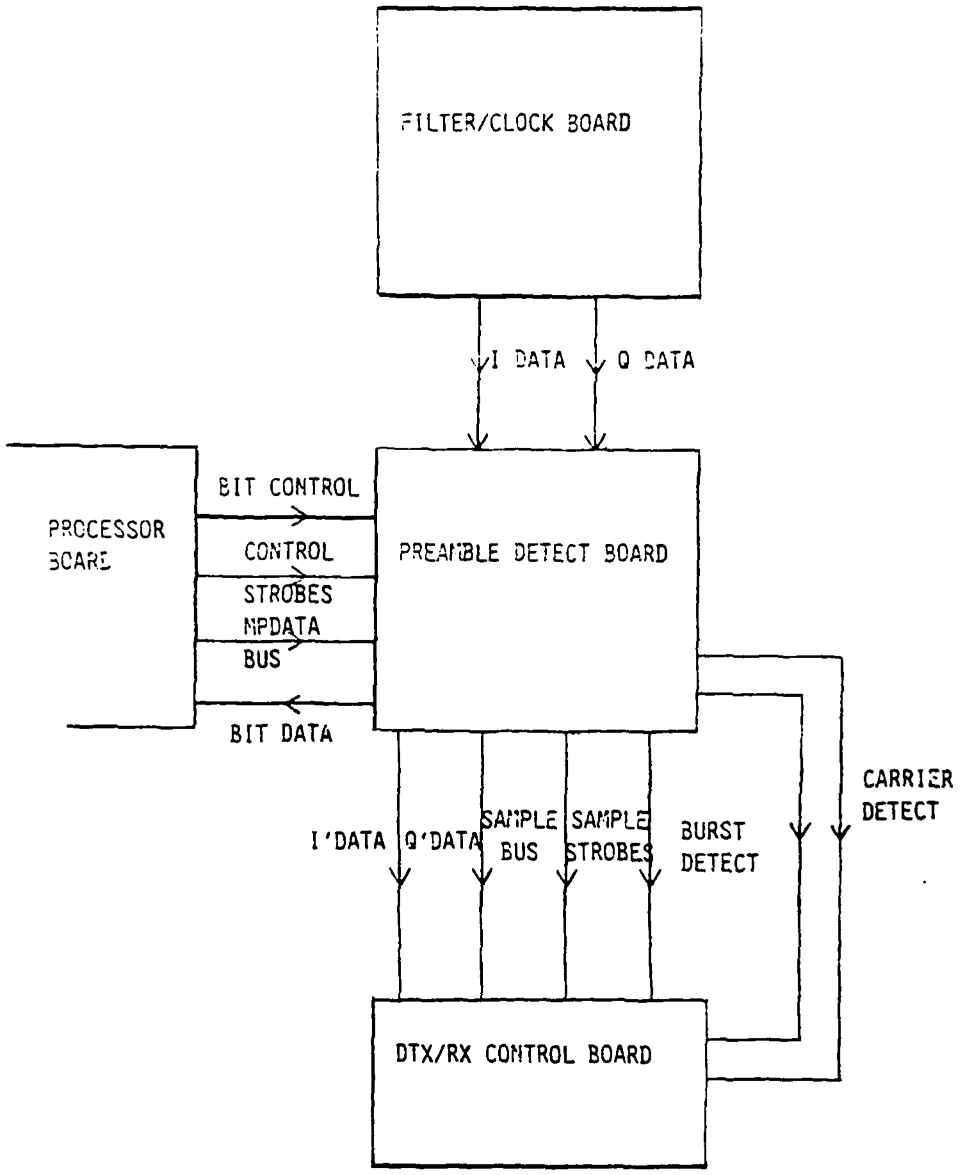


Figure 1. Preamble Detect Board System Configuration

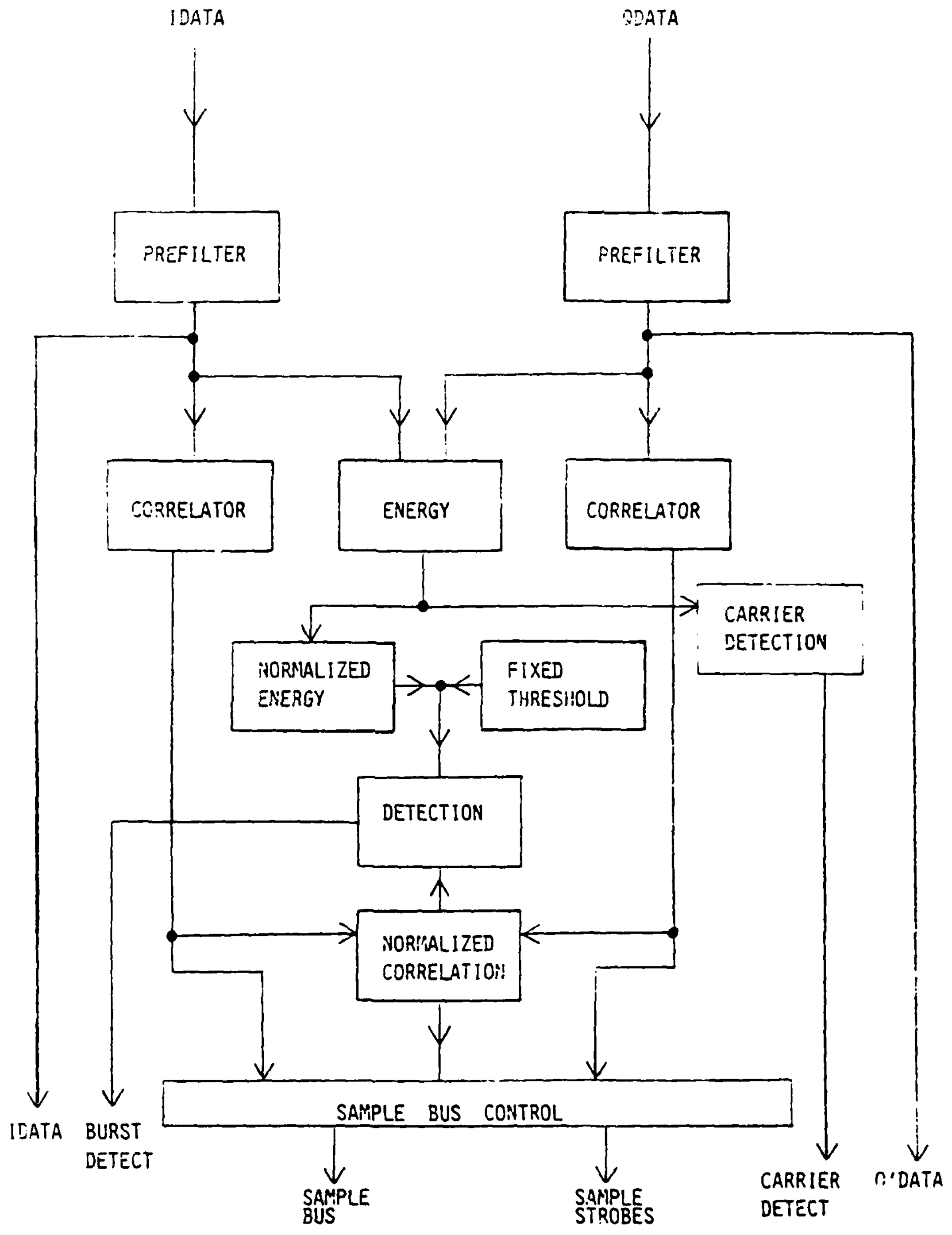


Figure 2. Data Processing Structure

Following detection, various samples used in the initial estimation of modem parameters are passed externally to the DTxRx board. Sample strobes are used to control passing of these parameters. In addition, the sample strobe occurring last (in time) notifies the DTxRx board that all estimates are available, and tracking will soon begin.

The processor board sends control and threshold information to the Preamble Detect board. It also initiates and evaluates built-in test functions.

PREFILTER OPERATION

In order to simplify implementation the system modulator and demodulator are baud rate independent. Figure 3 shows the system signal processing structure. Transmitted data at any symbol rate is generated by repeating transmission at the highest symbol rate. Following demodulation, it is necessary to demodulate data at the symbol rate used in modulation. (Symbol rate is a system parameter, controlled by the PSAT through the modem processor.) The process needed to obtain the receive data is an integrate and dump filter. The prefilter performs this operation.

The prefilter accepts partially demodulated data from the filter clock board. This data is the digitized waveform sampled at twice the highest symbol rate. Samples are divided into odd and even samples, as shown in Figure 4. During actual demodulation even samples fall on the center of the data impulse response, and odd samples fall on bit transitions.

The prefilter output consists of odd and even samples at the symbol rate in use. At the highest symbol rate output samples

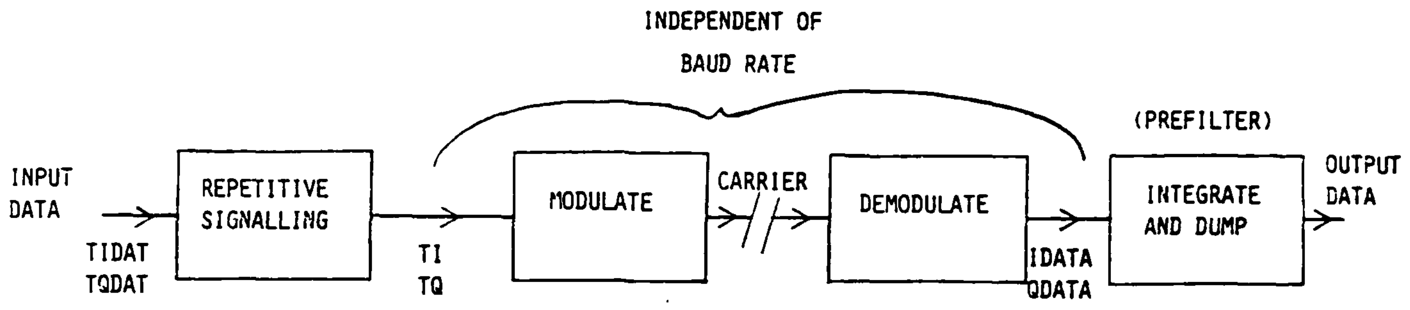


Figure 3. System Data Processing

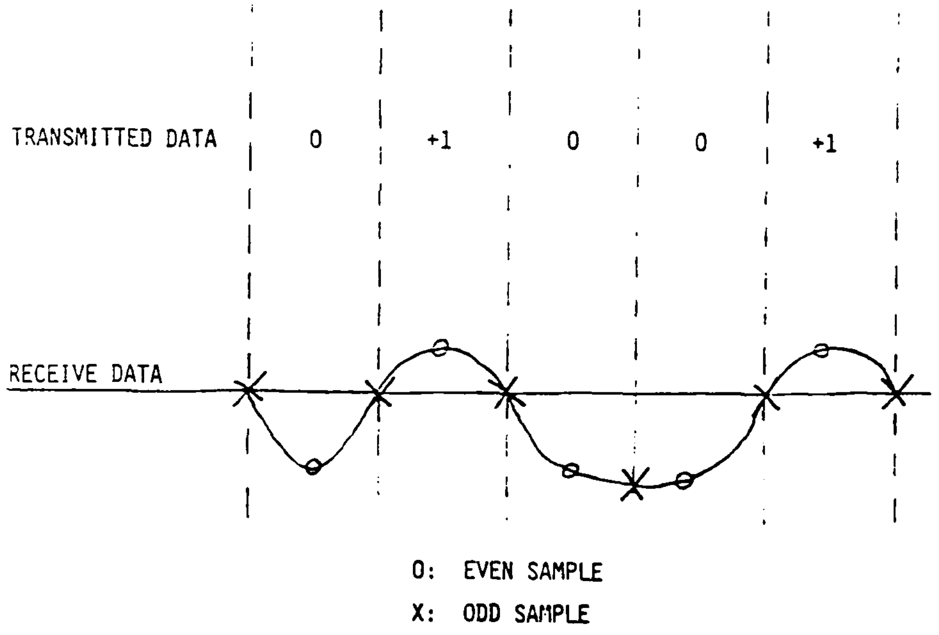


Figure 4. Receive Data Sampling, Highest Symbol Rate

are equivalent to input samples. At lower symbol rates output samples are averages of input even samples, grouped as in Figure 5. The definitions of output samples are:

$_aI[m]$: mth sample, ath symbol rate m=...,-1,0,1,...
m= odd:odd sample
m=even:even sample

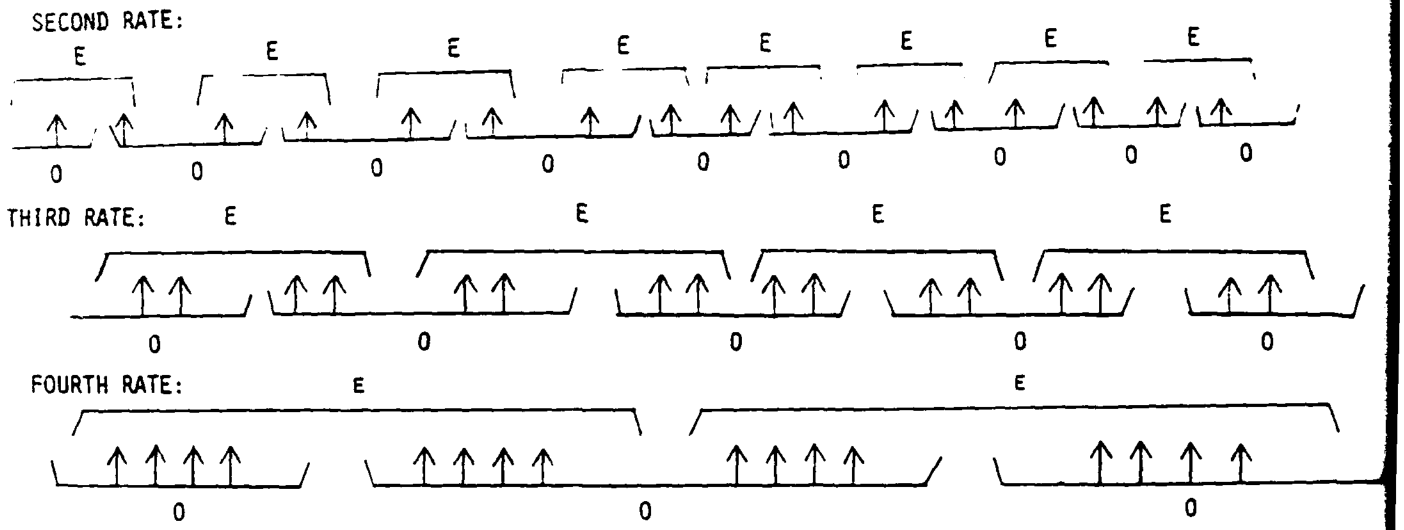
I[n] nith input sample, n=...,-1,0,1,...
n odd=odd sample
n even=even sample

Highest symbol rate: $_1I[m]+I[n+1]$

Second symbol rate : $_2I[m]=1/2(I[2m]+I[2m-2])$
$=1/2(_1I[2m]+_1I[2m(m-1)])$

Third symbol rate: $_3I[m]=1/4(I[4m]+I[4n-2]+I[4m-4]+I[4m-6])$
$=1/2(_2I[2m]+_2I[2(m-1)])$

Fourth symbol rate:
$_4I[m]=1/8(I[8m]+I[8m-2]+...+I[8m-12]+I[8m-14])$
$=1/2(_3I[2m]+_3I[2(m-1)]$



T: RECEIVE EVEN SAMPLES E: EVEN OUTPUT SAMPLES O: ODD OUTPUT SAMPLES

Figure 5. Prefilter Sampling

Note that each symbol rate can be determined from the average of the two previous even samples at the next higher symbol rate, except for the highest symbol rate. Figure 6 shows the prefilter structure for this operation.

Data relationships are defined by the clocks RX4-RX1. RX4 rising clock edges occur at twices the highest baud rate, RX3 at the second highest baud rate, etc. Figure 7 defines the

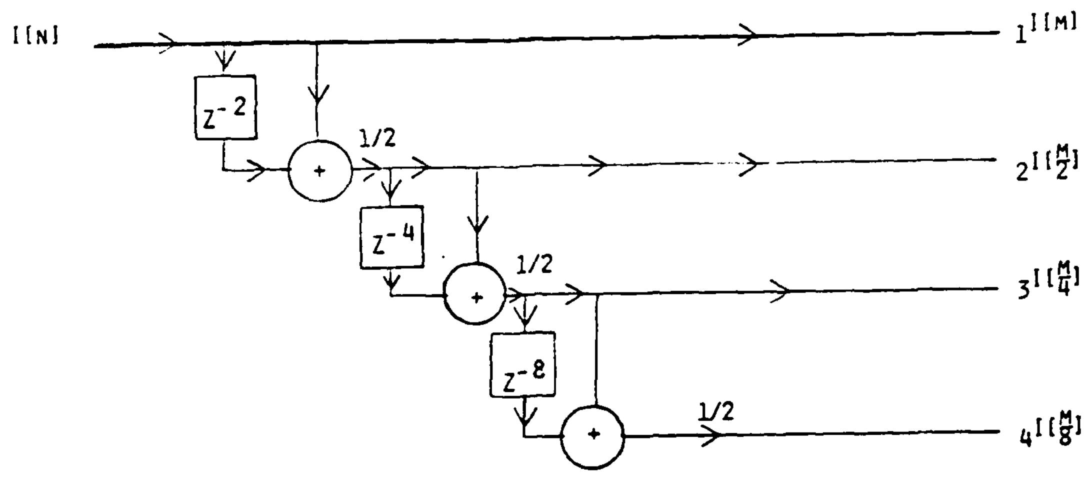


Figure 6. Prefilter Functional Structure

relationship of RX4 through RX1. Input samples are latched on the falling edge of RX4. Lower rate samples are defined on the falling edge of RX3 through RX1, to be used on the next rising edge.

Another prefilter function is to generate 2RXbaud (a clock at twice the baud rate) to be used internally and externally. 2RXbaud is simply a selection of RX4 (highest rate) through RX1 (lowest rate). The relationship of rate and the selection bits is shown in Table 1.Prefiltered data is presented to correlator and energy circuits. In addition, prefiltered data is fed to the digital demodulator on the DTX/RX Control Board. This data must be latched on the falling edge of 2RXbaud, to be used on the next rising edge (the opposite of normal procedure). Appendix II gives a complete list of signals used or generated by the prefilter. Figure 8 contains a block diagram of the prefilter sub-unit.

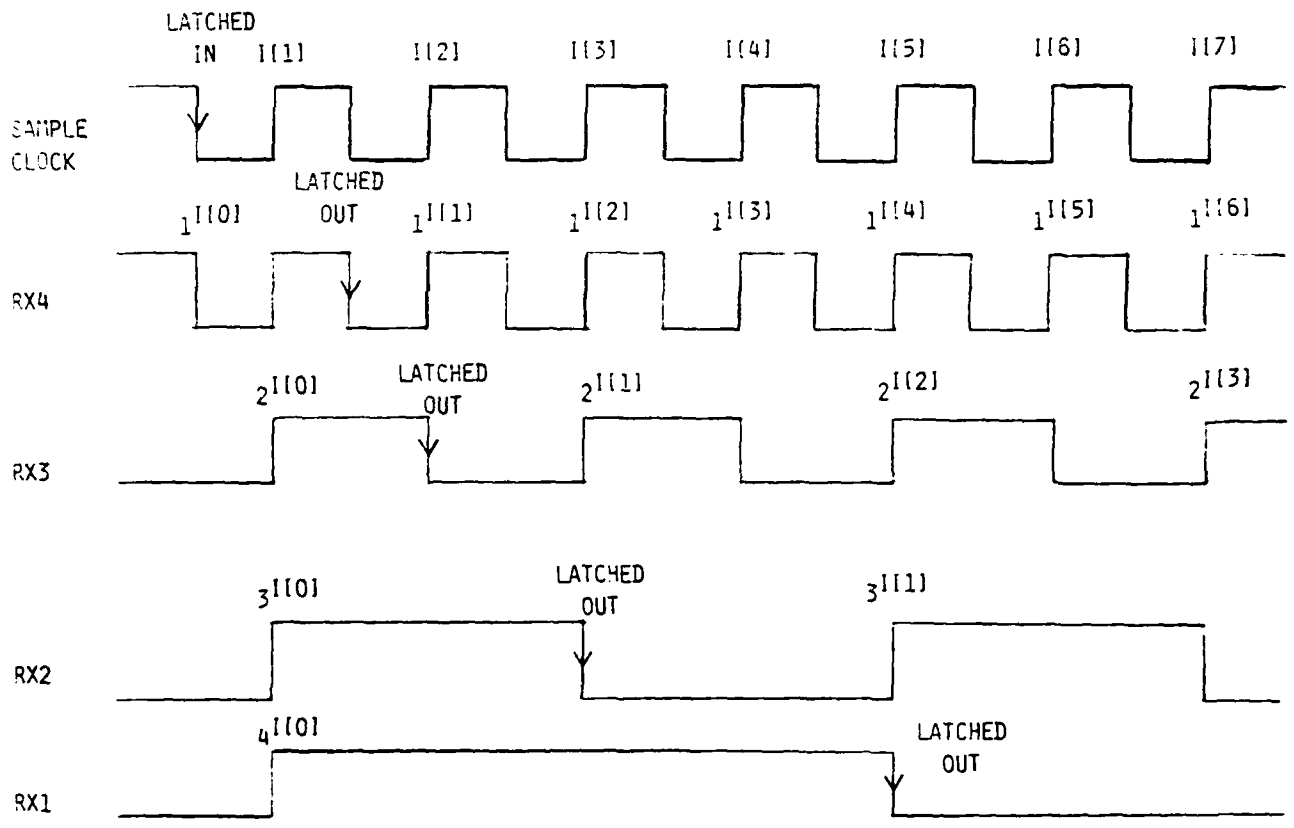


Figure 7. Receive Rate and Reference Clock Relationships

RXRATE	2RXBAUD	SYMBOL RATE	RATE
00	RX4	1544 KSPS	HIGHEST
01	RX3	722 KSPS	SECOND
10	RX2	386 KSPS	THIRD
11	RX1	193 KSPS	FOURTH

Table 1. Receive Rate Selector

The prefilter implementation is an open-loop adding of even samples of the even samples of the higher rates to generate lower rate symbols. This can be done easily using the MSI structure diagrammed in Figure 8. The previous even sample of the next higher rate is held in a latch. The new even sample is added to the previous even sample to generate the new symbol. Data is then selected using the rate control bits for internal and external processing.

The advantages of this implementation are speed and simplicity. The algorithm used is non-recursive, so temporary failure will not have long-term effects. Disadvantages are IC count and the large number of internal signal lines required.

CORRELATOR OPERATION

The correlators perform a coherent correlation of the filtered input data with a predetermined binary sequence over 32 samples. Correlation is performed independently on odd and even samples and on I and Q data channels.

Figure 9 outlines the basic structure. Correlation with selected sequence is performed on the receive data. The operation performed is:

$$c[n] = \sum_{i=-31}^{0} a_i x[n-2i]$$

where a_i is the correlation sequence, x[n] is the nth input sample, C[n] is the nth correlation. ($a_i=\pm1$). Correlation control should initialize and control any required functions. Correlation delays are shown in Figure 10.

The correlation function can be achieved using the structure in Figure 11. For details on the arithmetic operations, see appendix IV. Basically, data is fed bit by bit into one of four digital correlators. The data is accepted on the falling edge of 2RXbaud, then latched into a summer on the chip on the next falling edge of 2RXbaud. Output is then scaled and added with other outputs (or subtracted) as shown. The final result is converted to sign magnitude and latched on the second falling edge of 2RXbaud.

The control circuit must initialize the correlators by loading 1010 into the Mask register and the correlation sequence into the R register (through the B register). To reduce sensitivity to bit errors in loading the sequence, initialization will be repeated during the first 32 symbols of each burst, when the correlator load signal is active.

The detection sequences are stored in a PROM with one of two possible sequences depending on the detect sequence select controlled through the processor port.

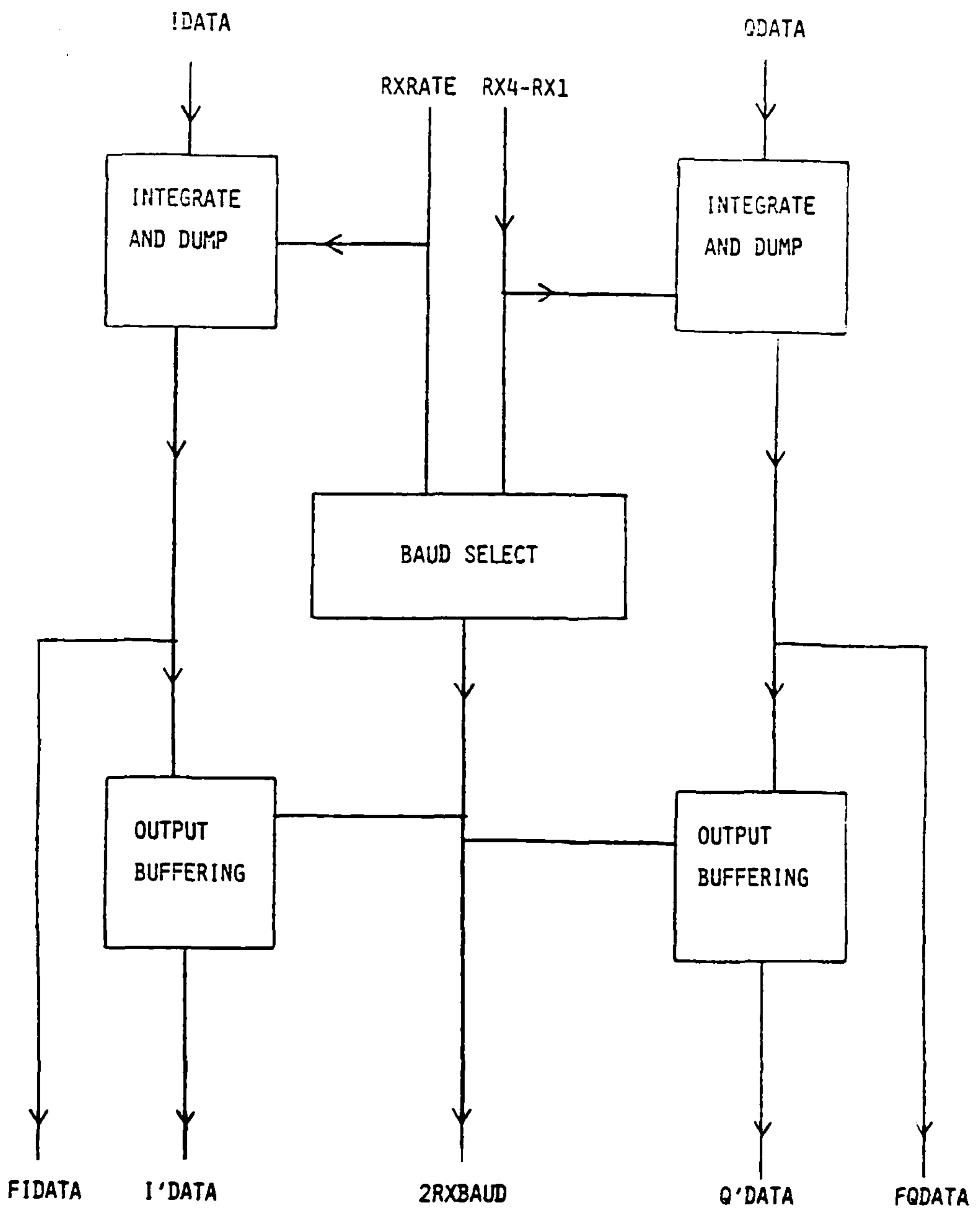


Figure 8. Prefilter Block Diagram

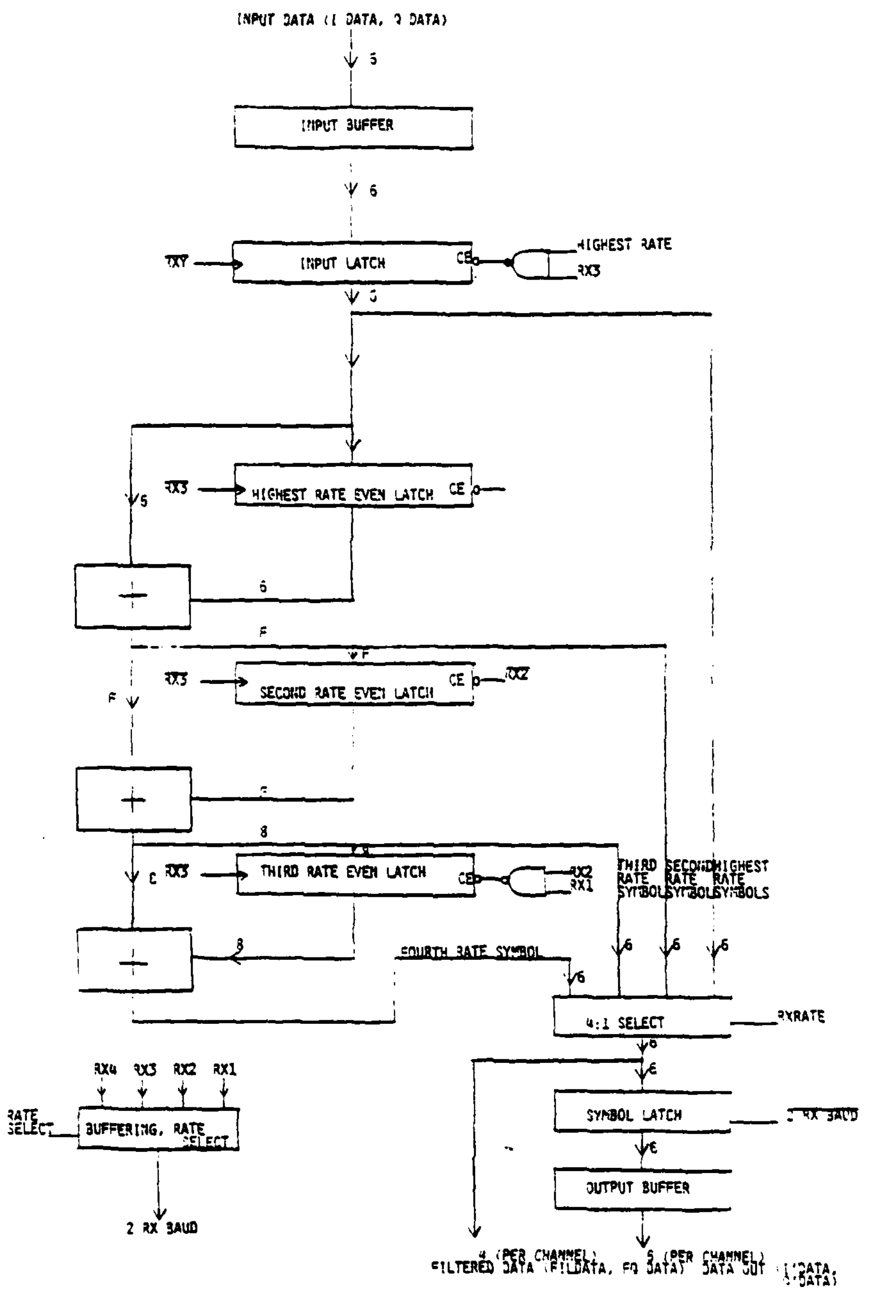
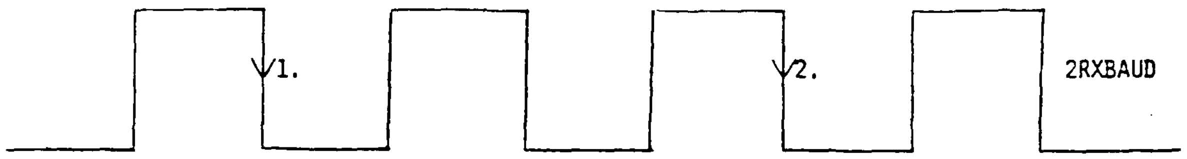


Figure 9. Prefilter Implementation



1. FIDATA, FQDATA SAMPLE N ACCEPTED BY CORRELATOR

2. ICORR, QCORR LATCHED FROM CORRELATOR

Figure 10. Correlator Processing Delay

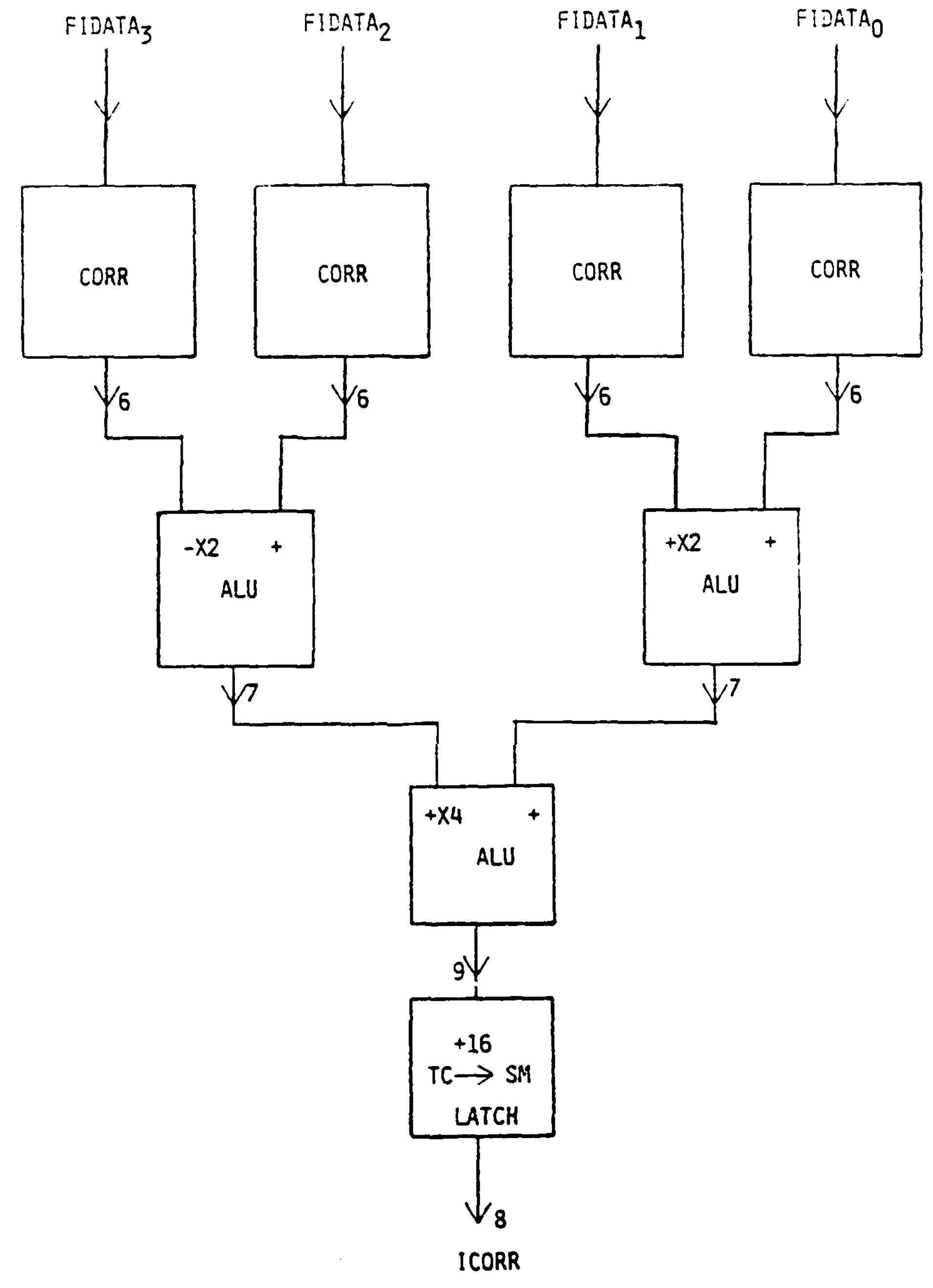


Figure 11. Correlation Implementation

ENERGY COMPUTATION

The energy of the input waveform must be computed over a 32 symbol window. The energy of a sequence is defined by:

$$E[n] = \sum_{i=-31}^{0} I^2[n-2i]+Q^2[n-2i]$$

In addition, the energy is to be normalized for further use by multiplying it by a scale factor, K. Carrier detection is also performed by comparing the energy to a threshold determined from the processor port. Figure 12 shows the energy circuit structure.

For detection purposes, a non-linearity is added to the energy computation (but not for carrier detection). If the energy is less than some value, the energy is fixed at that value. This is to minimize probability of detection for signals of very low applitude.

Figure 13 outlines the energy circuit implementation. Filtered data is latched on the falling edge of 2RXbaud. A 64-deep shift register is used to store samples. The sample outputted from the register is subtracted from the energy and the input sample is added to it (after squaring). Energy squared is stored separately for odd and even samples. Output energy is latched on the appropriate cycle.

Carrier detection is compared by comparing the energy with an eight-bit threshold fixed by the processor port. It may be desirable to expand this to 12 bits.

The algorithm used for energy determination is recursive. Each energy is predicated on the previous energy. Thus:

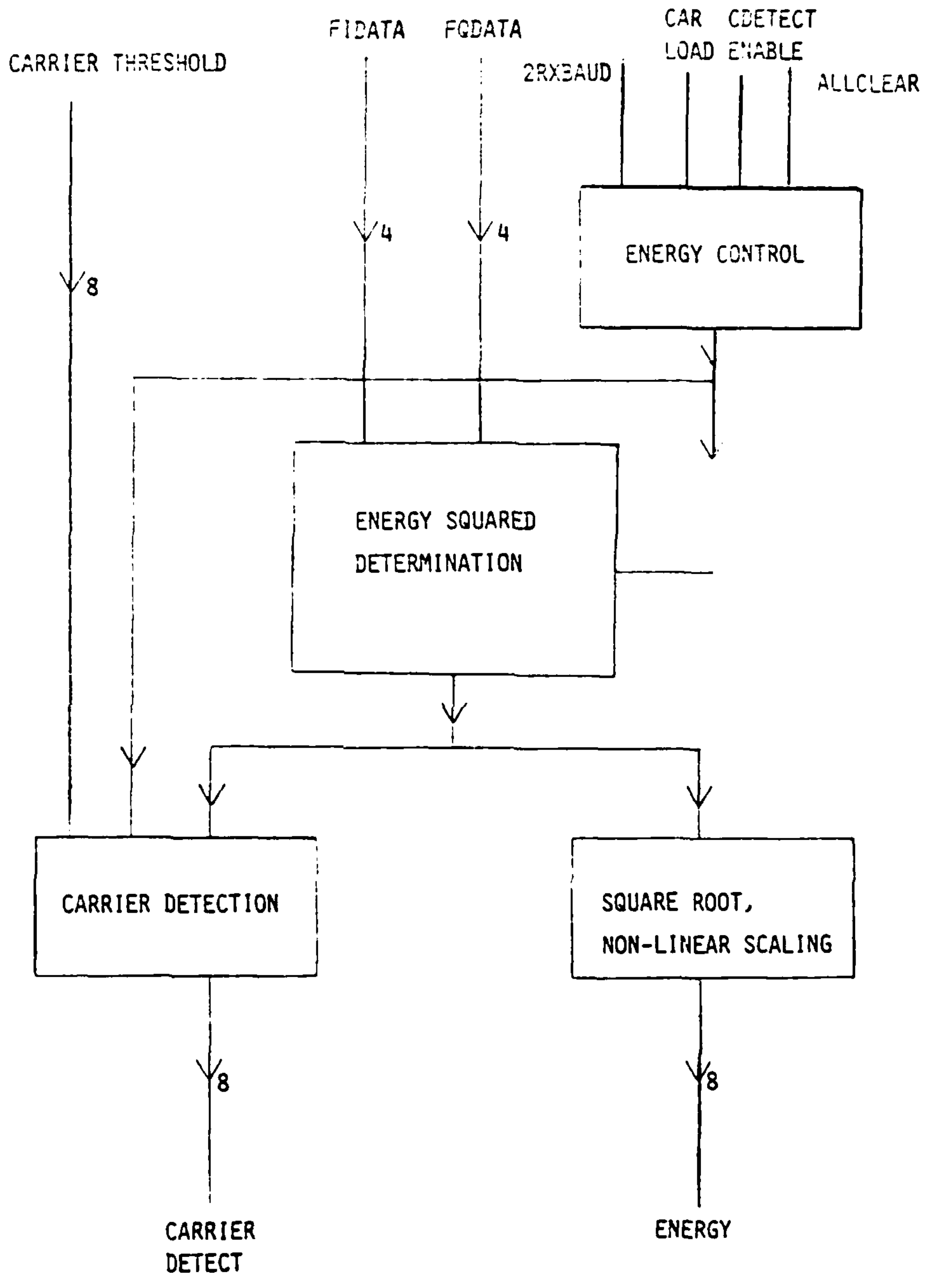


Figure 12. Energy Determination

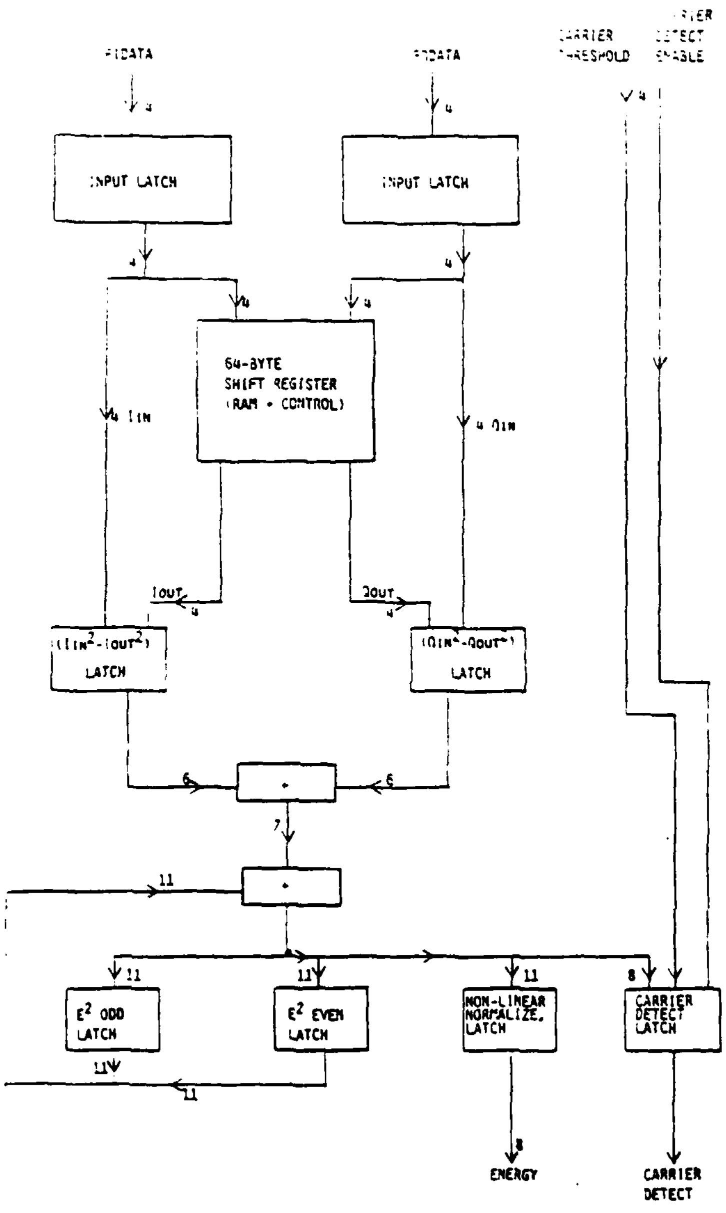


Figure 13. Energy Circuit Implementation

$$E[n]=E[n-2]+I^2[n]+Q^2[n]-I^2[n-64]-Q^2[n-64]$$

For this reason, the energy shift register and latches are zeroed whenever necessary. This can occur on power-on and at the start of each burst (at the same time the correlators are re-initialized). In addition, if burst detection is disabled, the energy (and correlators) are re-initialized.

The implementation of the shift register uses RAM, one for even samples, one for odd. Two 64 byte RAMs with identical addressing are used, one is written to while the other is read.

Note that the energy squared terms can be expressed in 11 bits even though the maximum value would require 12 bits. This is because the energy must always be a multiple of 2.

DETECTION FUNCTION The purpose of detection is to determine the start of a burst, then to determine the point where maximum correlation has occurred. Detection occurs when the normalized correlation exceeds the normalized energy (in CFAR) or the correlation exceeds a fixed threshold (fixed threshold). After detection occurs, the normalized correlation is compared with its previous value. When the previous correlation exceeds the present correlation, the detect done flag indicates the detection process is complete.

A burst is terminated when preamble reset occurs. This signal is used to end the burst, after which the detection process starts up again. Preamble reset forces BDTECT to go inactive. Control then can either force burst detect to remain inactive for the next 32 symbols (to prevent detection on the end of a burst) or allow it to go active at once.

Control also generates the signal which causes the correlator and energy units to be reloaded.

Figure 14 outlines the basic structure. First, the detection test is applied:

detection goes high if $NC[n] > NE[n]$ (CFAR)
or $NC[n] > Thr$ (Fixed Threshold)
$NC[n] = [Icorr^2[n] + QCorr^2[n]]^{1/2}$
detection done goes high if $NC[n] < NC[n-1]$ and
detection has gone high.

Figure 15 shows the numerical process involved.

In addition, the normalized correlation generated is passed to the sample bus control for use in estimation.

Figure 16 outlines the basic detection structure. A tristate bus controlled by the detection control is used with one comparator to generate the desired states. Flip-flops and gates are used to generate the various detection signals. Finally, a 1Kx8 PROM is used for the square root.

ESTIMATION PASSING

After detection, estimates of initial carrier parameters are generated by the DTX/RX control board. These estimates are generated using:

the maximum normalized correlation (EMAX) (for AGC)

the previous normalized correlation (EMAX-1)
(for bit timing)

the successive normalized correlation (EMAX+1)
(for bit timing)

the Inphase correlation of the max correlation (IMAX)
(for phase)

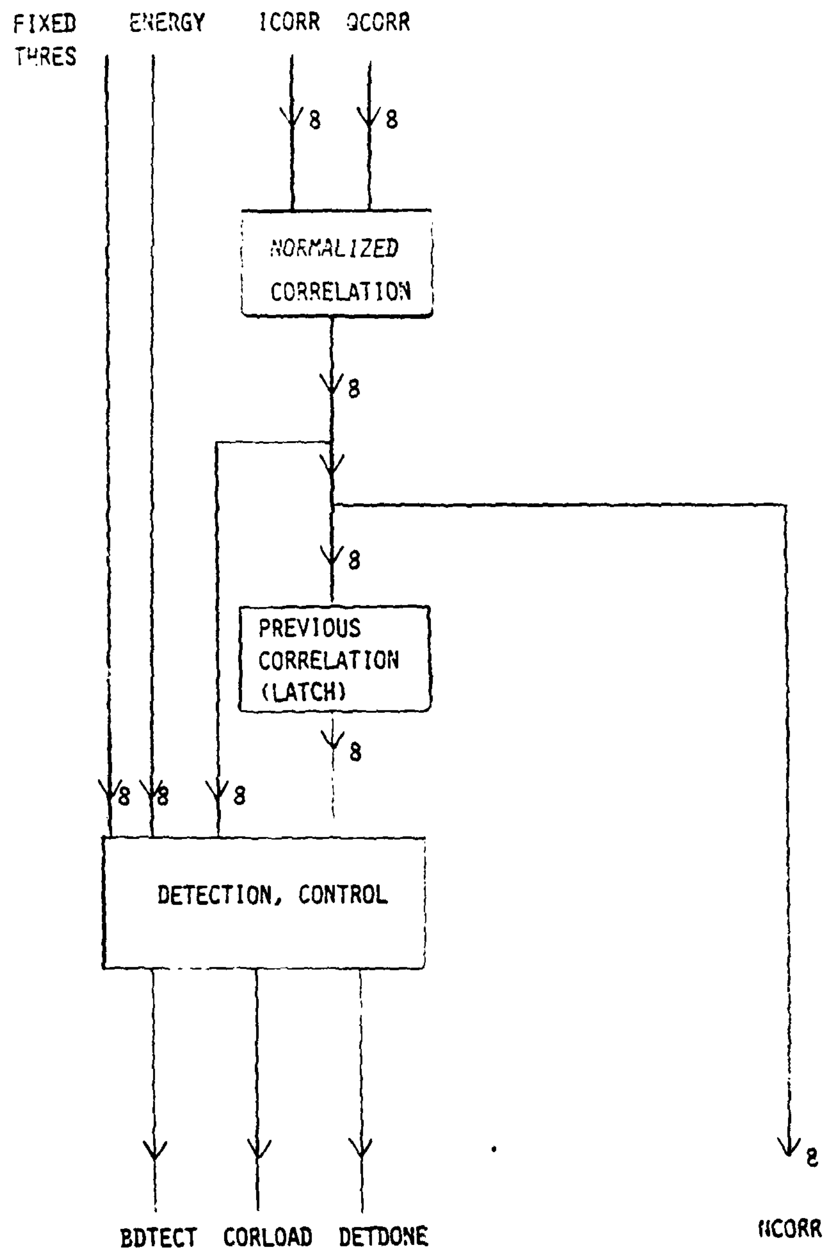


Figure 14. Detection Circuit Structure

the quadraphase correlation of the max correlation (QMAX)
(for phase)

The critical estimate is of the symbol timing. For this reason, these estimates should be prepared as soon as possible. Ohter estimates can follow logically.

All estimate values are sent out over an eight-bit sample bus. There are four strobes, one for EMAX and EMAX-1, plus one for each additonal sample. Before detection occurs, EMAX should

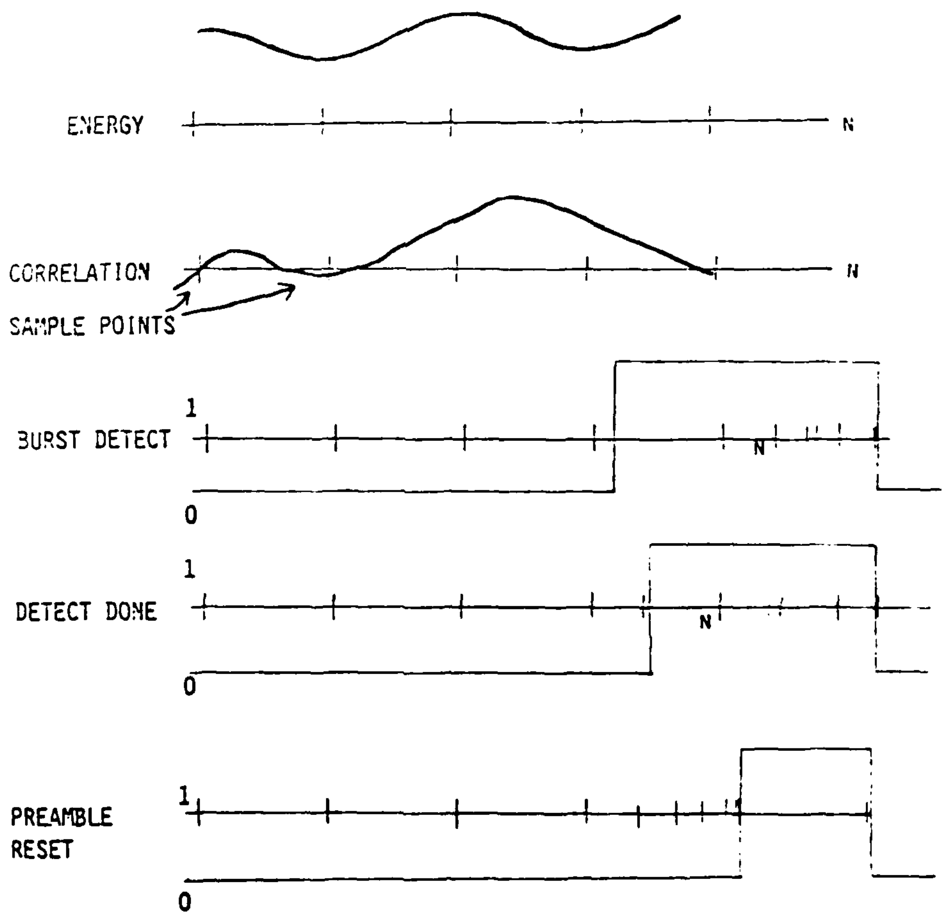


Figure 15. Detection Process

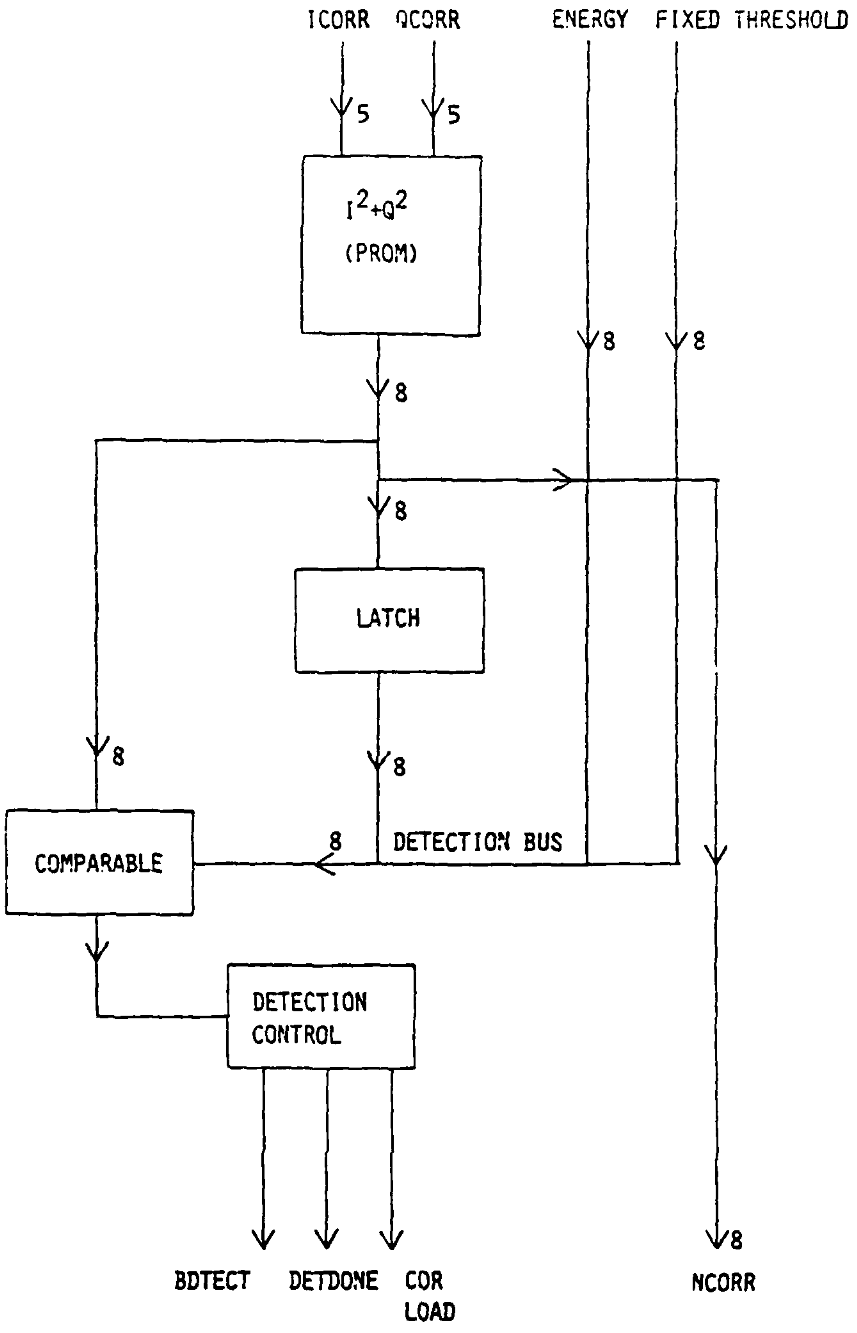


Figure 16. Detection Implementation

clock samples to the bus every cycle. This latches the present value into the EMAX register and the previous value into the EMAX-1 register on the DTX/RX control. Each such sample is a potential maximum, depending on following events. The procedure is continued on the following cycles, until detect done goes active. At this point EMAX+1 is available and, after this is sent, IMAX and QMAX are transferred. Following transfer of these samples, the bus and strobes are inactive until the burst is is terminated. Figure 17 shows the strobe structure.

Figure 18 shows the circuit design. The ICorr and QCorr samples are held following detection done. Strobes are generated as above. One-shots are used on the strobes, to make them go inactive 20ns (minimum) before 2RXbaud goes active. The actual bus signals are active following the rising edge of 2RXbaud, until the start of the next rising edge.

BUILT-IN-TEST

Built-in test used to perform signature analysis on one of 16 signals. Built-in test is controlled from the processor port. When internal signal generation is required, all inputs to the board are generated using an eight bit counter and PROMs. The PROMs are programmed to generate signals which exercise the board. The counters are clocked on bit clock, and synchronized by clearing them when bit sync is active.

The output signal, selected by a 16-1 multiplexer, is enabled onto the bit data line if the global signal BITDIS is inactive and the individual BIT enable signal is active (from the processor port).

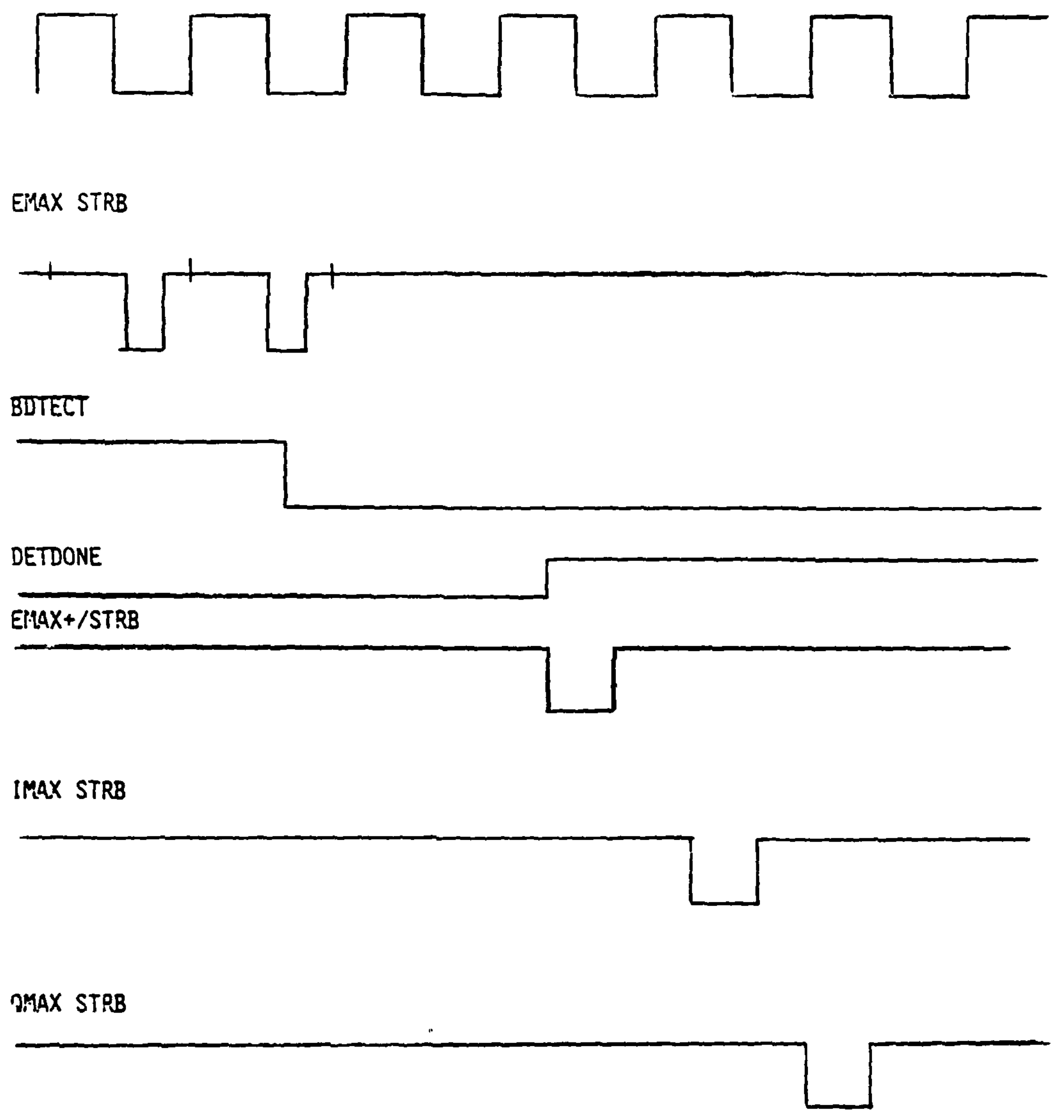


Figure 17. Sample Strobe Structure

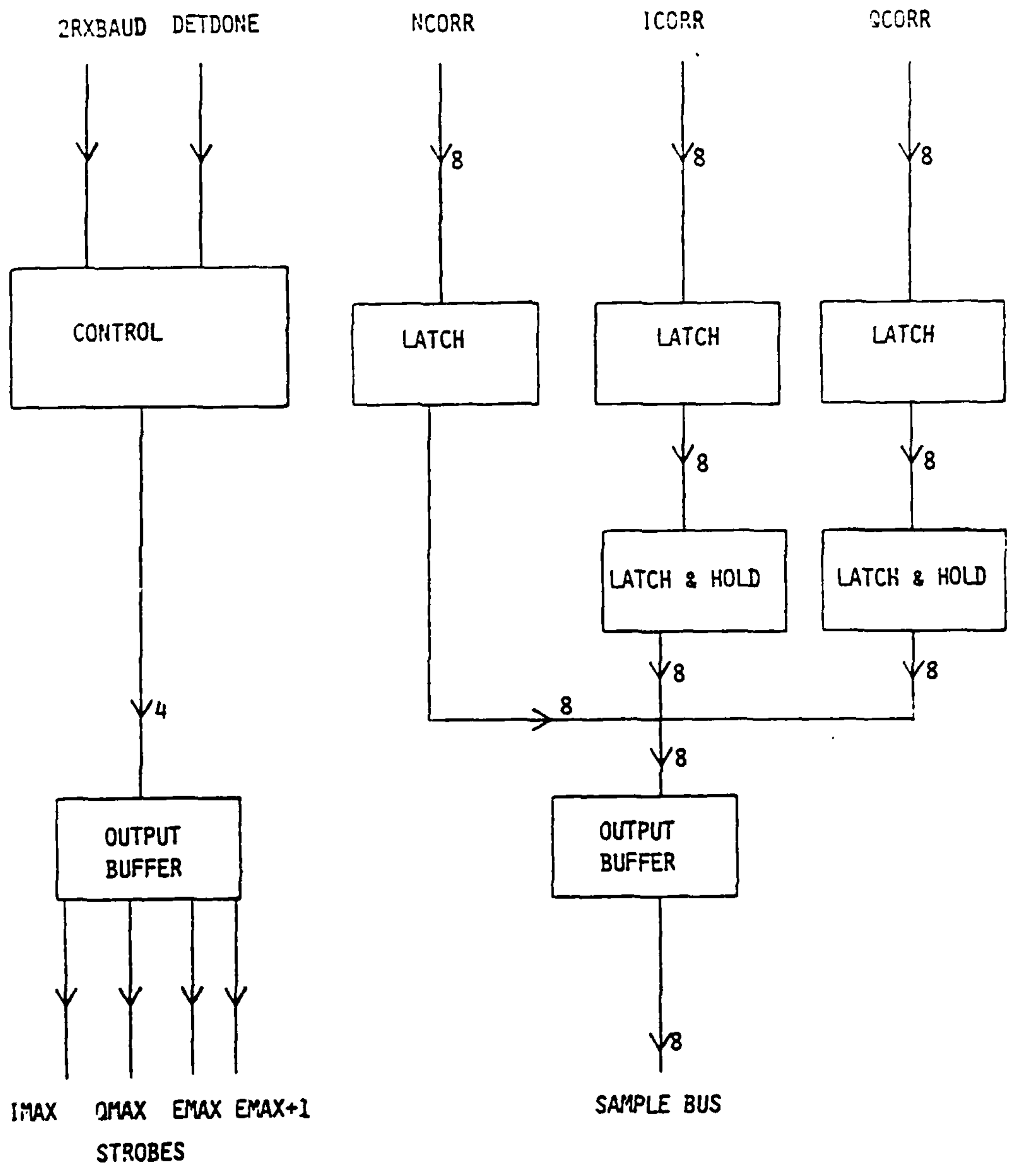


Figure 18. Estimation Passing Design

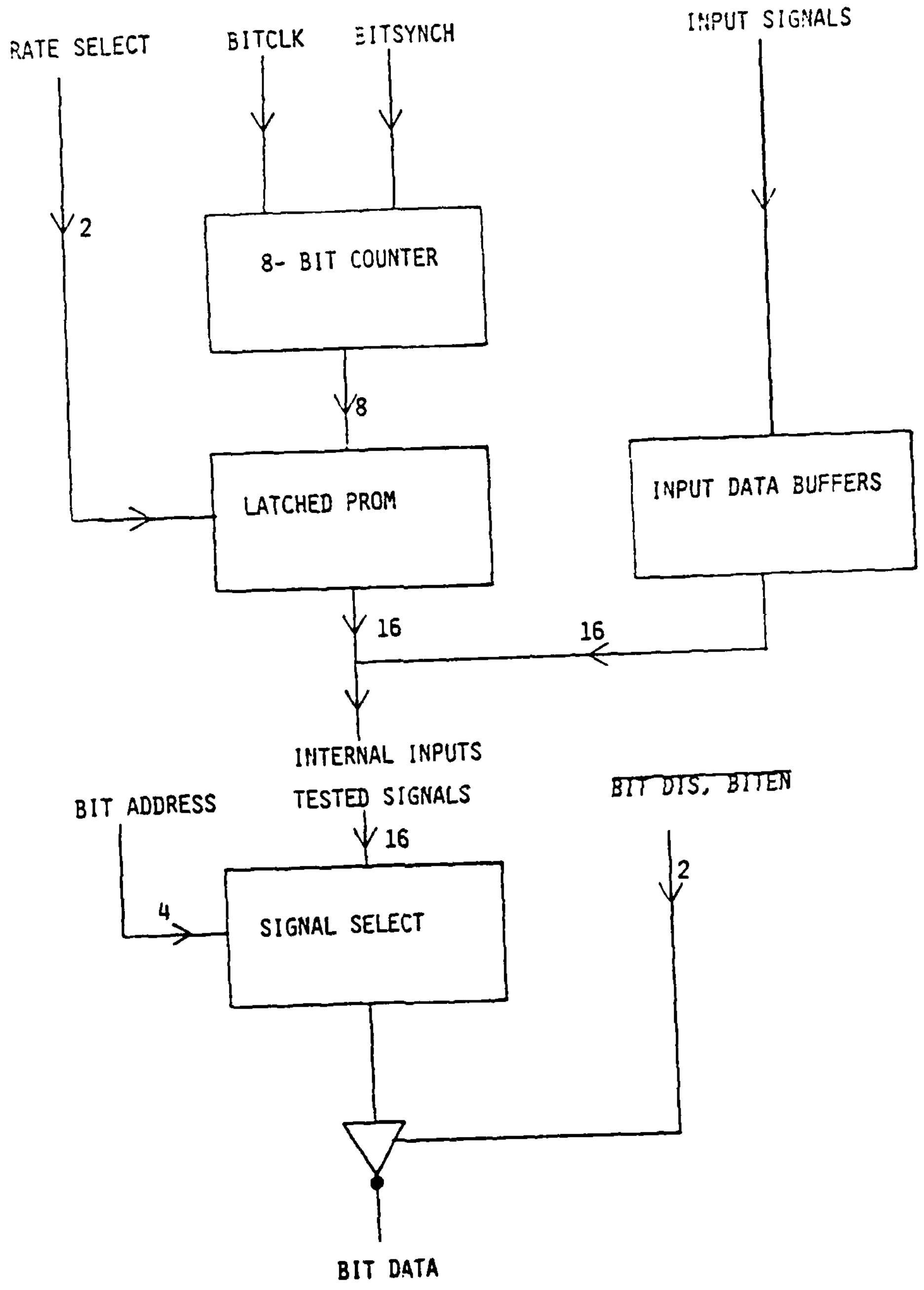


Figure 19. BIT Structure

Figure 19 shows the structure of the built-in test circuit.

PROCESSOR PORTS

The microprocessor controls the board by writing into one of four ports. The ports are each latched with the processor generated strobe appropriate for that port.

The control port is used to fix preamble length, preamble type, symbol rate, fixed threshold enable, and carrier detect enable. The threshold ports fix the fixed detection threshold and carrier detect threshold levels. The BIT port controls built-in test. Figure 20 shows the structure of the various ports.

PT	CE	DE	FTE	PL	SR
7	6	5	4	3-2	1-0

PT:	PREAMBLE TYPE
CE:	CARRIER DETECT ENABLE
DE:	DETECT ENABLE
FTE:	FIXED THRESHOLD ENABLE
PL:	PREAMBLE LENGTH (NOT USED)
SR:	SYMBOL RATE

MS THRESHOLD

CAR MS	THR MS
7-11	3-0

LS THRESHOLD

CAR LS	THR LS
7-4	3-0

CAR:	CARRIER DETECT THRESHOLD
THR:	FIXED THRESHOLD

BIT CONTROL

BOE	BDE	NOT USED	BIT ADDR
7	6	5-4	3-0

BOE:	BIT OUTPUT ENABLE
BDE:	BIT DATA ENABLE
BIT ADDR:	BIT ADDRESS

Figure 20. Processor Ports

Appendix I. Preamble Detect I/O Signals

Name	Bits	I/O	Srce/Dest.	Description
IDATA	6	I	Filter/Clock	Inphase receive
QDATA	6	I	Filter/Clock	quadraphase receive data
I'DATA	6	0	DTX/RX Control	Prefiltered inphase receive data
Q'DATA	6	0	DTX/RX Control	Prefiltered quadraphase receive data
SAMPLE	8	0	DTX/RX Control	Estimate sample bus
EMAX STRB	1	0	DTX/RX Control	Normalized correlation strobe
EMAX+1 STRB	1	0	DTX/RX Control	Last normalized correlation strobe
IMAX STRB	1	0	DTX/RX Control	Inphase correlation strobe
QMAX STRB	1	0	DTX/RX Control	quadraphase correlation strobe
Prerest Pre	1	I	DTX/RX Control	Terminates Burst
BDTECT	1	0	External	Indicates presence of a burst
CDTECT	1	0	External	Indicates presence of a carrier
MPBUS	8	I	Modem Processor	Microprocessor Bus
Precntlstrb	1	I	Modem Processor	Control Port Strobe
Prethrsstrb	1	I	Modem Processsor	Threshold Port Strobe
Presibstrb	1	I	Modem Processor	Built-in Test Port Strobe
RXRATE	2	0	External	Receive symbol rate
PRELEN	2	0	External	Preamble Length (not used)
RX4-RX1	4	I	Filter/Clock	Receive timing clocks
2RXBAUD	1	0	DTX/RX Control	Twice receive baud time clock
24.7MHz	1	I	Filter/Clock	24.7 MHz Reference (not used)
BITCLK	1	I	Modem Processor	Built-in-test clock
BITSYNCH	1	I	Modem Processor	Built-in-test sync
BITDIS	1	I	Modem Processor	Built-in-test diable
BITDATA	1	0	Modem Processor	Built-in-test data
Prelsthrstrb	1	I	Modem Processor	Low order threshold strobe

Appendix 2. Prefilter I/O Signals

Name	BITs	I/O	Srce/Dest.	Description
IDATA	6	I	Filter/Clock	Inphase receive data
QDATA	6	I	Filter/Clock	quadraphase receive data
RXRATE	2	I	Control Port	Receive symbol rate
RX$RX1	4	I	Filter/Clock	Receive timing clocks
FIDATA	4	0	ICorrelator, Energy	filtered Inphase data
FQDATA	4	0	QCorrelator, Energy	Filtered quadraphase data
I'DATA	6	0	DTX/RX Control	Prefiltered inphase receive
QDATA	6	0	DTX/RX Control	Prefiltered quadraphase receive data
2RXbaud	1	0	internal, DTX/RX Control	Twice receive baud time clock

Appendix III. Correlator I/O Signals

Name	BITs	I/O	Srce/Dest.	Description
FIDATA	4	I	Prefilter	Filtered inphase data
FQDATA	4	I	Prefilter	Filtered quadraphase data
PT	1	I	Processor	Preamble type
Corrld	1	I	estimation	correlator sequence load
Allclear	1	I	external	power-on reset
2RXbaud	1	I	prefilter	twice baud time clock
ICorr	9	0	detection estimation	inphase data correlation
QCorr	9	0	detection, estimation	quadraphase data correlation

Appendix IV. Correlation Mathematical Operation

Detection mode: In detection mode, the correlators determine the coherent correlation of the input data with a fixed sequence:

$$C[n] = \sum_{i=-31}^{0} a_i X[n+2_i]$$

Where C[n] is the correlation of the nth sample
X[n] is the nith input sample
a_i is the ith bit of the correction sequence

a_i =1 or 1, fixed for each i
implemenation limitations require
$-15\ 1/2 \leq X[n] \leq 15\ 1/2$ in increments of 1.

Note that X[n] is an actual discrete value. It will be represented in actual hardware by Y[n], a two's complement notation in 4 bits offset from X[n] by -1/2. (X[n]=Y[n]=+1/2).

Implementation with discrete digital correlators is required. Each correlator performs the operation:

$$D[n] = \sum_{i=-31}^{0} m_i\ (\overline{R_i \oplus Y[n+i]})$$

Where Y[n], r; m; are digital one-bit values (0,1). This function must be used to generate the coherent correlation referred to above.

To start, in two's component notation the bits of Y[n] are defined from the relation:

$$Y[n]=y_o[n]+2y,[n]+4y_2[n]-8y_3[n]$$

where ym is the nth digital bit of Y. Therefore

$$C[n] = a_i[y_o[n+2_i+2y_1[n-2_i+4y_2[n+2_i]$$
$$-8y_3[n+2_i]+1/2]$$
$$=C_o[n]+2C_i[n]=4C_2[n]-8C_3[n]+K$$

where $C_m\ [n]=C[y_m[n]=\ a_iY_m[n+2_i]$

$K = a_i$

We now need a relationship between C[n] and D[n].

If m_i is defined by: $m_i=0;\ i=-1,-3,\ldots,-61,-63$

$m_i=1,\ i=0,-2,\ldots,-60,-62$

we get $D[n] = \sum_{i=63}^{0} m_i(\overline{r_{2i} \oplus y[n+2_i]})$

We want to find R[n], where

$C[n]=R[n]+D[n] \quad m=0,3$

If R[n] = R (Independent of n), the correlators can be used.

$$R = \sum_{i=-31}^{0} [\overline{R_{2i} \oplus y[+2_i]} - a_i y[n+2_i]]$$

for each term:

a_i	$y[n]$	$a_i y[n+2_i]$	r_{2i}	$r_{Qi}+y[n]$	$r_{2i}+y[n]-a_i y[n]$
+1	0	0	1	0	0
+1	1	1	1	1	0
-1	0	0	0	1	1
-1	1	-1	0	0	1

$$R = \sum_{i=-31}^{0} \frac{a_i-1}{2}$$

combining the above results:

$$C[n]=D_o[n]+2D_i[n]+4D_2[n]-8D_3[n]+K-R$$
$$=D_o[n]+2D_i+4D_2[n]-8D_3[n]+16$$

with the relations for r_i and m_m as defined above.

This result justifies use of digital correlators in detection. In detection mode such a correlation must be computed.

Appendix V. Energy I/O Signals

Name	Bits	I/O	Srce/Dest.	Description
FIDATA	(4)	(I)	(Prefilter)	(Filtered Inphase data)
FQDATA	4	I	Prefilter	Filtered quadraphase data
CTHRES	8	I	Processor Port	Carrier detect threshold
2RXbaud	1	I	Prefilter	Twice baud time clock
Corld	1	I	Estimation	Correlator load
COE	1	I	Processor Port	Carrier detect enable
ALLCLEAR	1	I	External	Power-on reset
CDTECT	1	0	External	Carrier detect
Energy	8	0	Detection	Carrier energy

Appendix VI. Detection I/O Signals

Name	Bits	I/O	Srce/Dest.	Description
Fthrs	8	I	Processor port	Detection Fixed threshold
Energy	8	I	Energy	Carrier energy
ICorr	8	I	Correlator	Inphase correlation
BDTECT	80	internal,external		burst detect
Corrld	1	0	internal	correlator load
Detdone	1	0	internal	detection process done
NCORR	8	0	estimation	normalized correlation
2RXbaud	1	I	Prefilter	twice rate clock
Allclear	1	I	external	power-on reset
FTE	1	I	Processor port	fixed threshold enable
DE	1	I	Processor port	detect enable

Appendix VII. Estimation I/O Signals

Name	Bits	I/O	Srce/Dest.	Description
2RXbaud	1	I	Prefilter	twice receive baud clock
Detdone	1	I	Detection	detection done
NCORR	8	I	Detection	normalized correlation
ICORR	8	I	Correlation	Inphase correlation
QCORR	8	I	Correlation	quadrature Correlation
Allclear	1	I	external	Power-on Reset
1MAX STRB	1	0	external	Inphase max strobe
QMAX STRB	1	0	external	Quadrature max strobe
EMAX STRB	1	0	external	Normalized max strobe
EMAX+1 STRB	1	0	external	Normalized max+1 strobe
Sample bus	8	0	external	estimation sample bus

Appendix VIII. Bit I/O Signals

Name	Bits	I/O	Srce/Dest.	Description
RXRATE	2	I	Prefilter	Rate Select bits
BITCLK	1	I	External	BIT clock
BIT SYNCH	1	I	External	BIT synchronization pulse
BITDIS	1	I	External	BIT disable
BOE	1	I	Processor Port	BIT output enable
BDE	1	I	Processor Port	BIT data enable
BITADR	4	I	Processor Port	BIT address enable
BITDATA	1	0	External	BIT data
IDATA	6	I	External	Inphase data
QDATA	6	I	External	Quadrafure data
Preresetpre	1	I	External	Preamble reset
RX4-RX1	4	I	External	Receive reference clocks

Appendix IX. Processor Port I/O Signals

Name	Bits	I/O	Srce/Dest.	Description
MPBUS	8	I	External	data bus
Precntl strb	1	I	External	Preamble control strobe
Pre thrs strb	1	I	External	Preamble threshold strobe
Pre ls thrstrb	1	I	External threshold	Preamble low order
Pre bit strb	1	I	External	Preamble control strobe
PT	1	I	correlator	Preamble type
CDE	1	0	Energy	Carrier detect enable
DE	1	0	Detection	Detect enable
FTE	1	0	detector	fixed threshold enable
PL	2	0	External	Preamble length
RXRATE	2	0	Prefilter, external	Receive symbol rate
Cthr	8	0	Energy	Carrier detect threshold
Fthr	8	0	detection	Fixed detection threshold
BOE	1	0	BIT	BIT output enable
BDE	1	0	BIT	BIT detect enable
Bit addr	4	0	BIT	BIT data address

X. Basic Operation

The preamble detect board performs detection and estimation functions. Each burst is transmitted with a fixed 32 symbol preamble at the start. The preamble is the same on I and Q channels.

On the receive side, the preamble sequence is correlated on each channel with the incoming data whenever a burst is to be searched for. The preamble is selected to have low side lobes, and a low probability of detecting on data paterns. The energy of the signals within the same window is also computed. Detection occurs when the correlation exceeds the energy multiplied by a constant factor. the receive phase will not be correct; so the correlation is normalized by taking:

$(\lbrack n\rbrack = \quad I^2\lbrack n\rbrack+Q^2\lbrack n\rbrack)$. Since both

I[n] and Q[n] have the same data, phase terms are removed. Detection occurs when

$$\frac{C[n]}{E[n]} > \text{Thr.}$$

This is called CFAR (Constant False Alarm Rate). Alternately, detection can be performed with a fixed threshold:

$$C[n] > \text{FThr}$$

After detection, carrier parameters can be estimated from the various correlations generated, based on the maximum value of the correlation. These are:

BIT timing: from NC[n] /NC[n]

Phase: from $\tan^{-1}(Q[m]/I[m])$

Amplitude: from NC[m]

where NC[m] is the maximum value of the normalized correlation of the sequence.